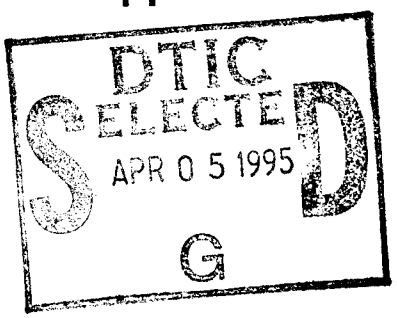
Task/Subtask IV02.1
CDRL Sequence A014–004
16 January 1995

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

Technical Papers: Using Process to Integrate Software Engineering Environments

Contract No. F19628–93–C–0129
Task IV02 – Megaprogramming Transition Support



Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731–2116

Prepared by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

Task/Subtask IV02.1
CDRL Sequence A014–004
16 January 1995

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

Technical Papers: Using Process to Integrate Software Engineering Environments

Contract No. F19628–93–C–0129
Task IV02 – Megaprogramming Transition Support

Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731–2116

Prepared by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

REPORT DOCUMENTION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1/16/95	Informal Technical Report

4. TITLE AND SUBTITLE
Using Process to Integrate Software Engineering Environments

5. FUNDING NUMBERS
F19628-93-C-0129

6. AUTHOR(S)
Dr. Richard L. Randall, Loral Federal Systems - Gaithersburg
William H. Ett, Loral Federal Systems - Gaithersburg

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

8. PERFORMING ORGANIZATION REPORT NUMBER
A014-004

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Electronic Systems Center/ENS
Air Force Materiel Command, USAF
5 Eglin Street, Building 1704
Hanscom Air Force Base, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
N/A

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Cleared for Public Release, Distribution is Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A layered architecture is fundamental to the megaprogramming approach being followed by the Air Force/STARS Demonstration Project, as engineers develop the Space Command and Control Architectural Infrastructure application for NORAD and USSPACECOM. This layering strategy also applies to the Software Engineering Environment (SEE) being used to support the application development. This paper focuses on one particular SEE layer: a layer that leverages one of an organization's most valued assets - its process - and uses it as a basis for SEE integration. The activity to produce and use the PSE has yielded useful lessons learned which can guide other organizations as they pursue their own process-driven SEEs.

14. SUBJECT TERMS
Software Engineering Environment (SEE), Process, Integration, Architecture, Megaprogramming

15. NUMBER OF PAGES
66

16. PRICE CODE
N/A

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

Preface

This document was developed by the Loral Federal Systems - Gaithersburg, located at 700 North Frederick Avenue, Gaithersburg, MD 20879. Questions or comments should be directed to Dr. Richard L. Randall at 719-554-6597 (Internet: randallr@lfs.loral.com).

Presenters: Dr. Richard Randall, William Ett (Loral Federal Systems)
Title: Using Process to Integrate Software Engineering Environments
Track: 7 - Architectures
Day: Tuesday, 4/11/95, 10:30 AM
Keywords: SEE, Software Engineering Environments, Process, Integration, Architecture, Megaprogramming

USING PROCESS TO INTEGRATE SOFTWARE ENGINEERING ENVIRONMENTS

1. INTRODUCTION

Modern software engineering principles call for separating systems into encapsulation layers that can be specified, designed, and implemented largely independently from one another. Such layers help engineers:

- Gain intellectual control of a system by dividing it into meaningful abstractions,
- Reduce the complexity of building the system by dividing it into separable pieces that can be built independently by a relatively small number of specialists (or perhaps procured commercially),
- Maintain the system more economically by localizing the impact of most changes, and
- Evolve the system more easily by allowing replacement of whole layers of functionality.

A layered architecture is fundamental to the megaprogramming approach being followed by the Air Force/STARS Demonstration Project, as engineers develop the Space Command and Control Architectural Infrastructure application for NORAD and USSPACECOM. The layering strategy also applies to the software engineering environment (SEE) being used to support the application development.

This paper focuses on one particular SEE layer: a layer that leverages one of an organization's most valued assets - its process - and uses it as a basis for SEE integration. This layer, termed the Process Support Environment (PSE), conceived by Loral Federal Systems[1] on behalf of the STARS program, is being used on the Air Force/STARS Demonstration Project SEE, in conjunction with a major Air Force priority to establish and support a megaprogramming product-line process.

[1] Formerly IBM Federal Systems Company

The activity to produce and use the PSE has yielded useful lessons learned which can guide other organizations as they pursue their own process-driven SEEs. The experience also motivates future productization in this area: a customizable process overlay that can be added to orchestrate an organization's existing SEE. The potential exists to add process integration to existing SEEs with modest investment - largely consisting of well-defined instantiation procedures.

To dramatize the relevance and timeliness of these results, the April, 1994 STSC Report on SEE Technology [STSC94] identified seven key technology areas that have received insufficient attention to date and in which the industry now seems postured to make significant progress. Of these seven areas, the top three were process modeling, process definition, and process enactment and enforcement. The PSE encapsulation layer discussed in this paper addresses all three of these areas. The layer also provides a tailorable metrics instrumentation framework to help the organization assess and improve their process.

The remainder of the paper is organized as follows:

- Section 2, **Context**, provides background on the STARS Program and the Air Force/STARS Demonstration Project;
- Section 3, **Process-Based SEE Integration**, elaborates the notion of encapsulation layers, discusses its relevance to SEE integration, and offers a model of the PSE encapsulation layer;
- Section 4, **The Air Force/STARS PSE: Experience To Date**, describes how the Demonstration Project PSE is currently constructed and the extent to which it adheres to the PSE model provided in Section 3, and summarizes our experience to date in developing, transitioning, and improving the PSE; and
- Section 5, **Conclusion**, presents our priorities for continuing the evolution of both the implementation of the PSE and the underlying ideas.

The reader is invited to refer to a closely related paper also contained in these proceedings, "Integrating a SEE for Megaprogramming: Lessons Learned" [Randall95], which discusses the Demonstration Project's SEE integration experience from a broader perspective.

2. CONTEXT

2.1 THE STARS PROGRAM

The ARPA STARS program is a technology development, integration and transition program to demonstrate a process-driven, domain specific, reuse-based approach to software engineering that is supported by appropriate tool and environment technology - an approach referred to as "megaprogramming".

Megaprogramming

Megaprogramming is a product-line (family of systems) approach to the creation and maintenance of software intensive systems. It is characterized by the reuse of

software life-cycle assets within a product-line including common architecture, models and components. Megaprogramming also includes the definition and enactment of disciplined processes for the development of applications and the evolution of the product-line as a whole. Finally, megaprogramming calls for automated support for the process via advanced software engineering environment (SEE) capabilities and their integration among those tools. For a more in-depth introduction to the STARS program and the notions of megaprogramming, please refer to [Trimble94].

Demonstration Projects

The STARS program mission is to accelerate the transition to the megaprogramming paradigm. Key vehicles for bringing this about are the three STARS Demonstration Projects - one with each of the three services (Air Force, Army, and Navy) - which are currently engaged in applying the principles of megaprogramming to real systems. The major objectives for each of the projects are:

- Apply megaprogramming principles to the development of software for an actual DoD application, to establish the credibility of the approach.
- Collect and document experience about the benefits and costs of megaprogramming as well as the effectiveness of the specific tools and techniques used on the project, to help other organizations plan for and implement similar approaches.
- Transition to the Demonstration Project's parent organization, to establish the capability to apply megaprogramming to other applications in their product-line.

2.2 THE AIR FORCE/STARS DEMONSTRATION PROJECT

The Air Force's Demonstration Project was identified in 1992: the Space Command and Control Architectural Infrastructure (SCAI) project, to be managed by AFSPC's[2] Space and Warning Systems Center[3] (SWSC) at Peterson AFB, Colorado. Loral Federal Systems[4], one of three STARS prime contractors, was paired with the Air Force for the project.

The SWSC is responsible for the maintenance of the application software for the C^2 centers at the Cheyenne Mountain Air Force Station (CMAS) - which are responsible for national attack warning/assessment and space surveillance/defense/control. A large number of mission-critical systems are involved, with a high annual maintenance cost.

[2] Air Force Space Command

[3] Effective Februrary, 1995, the SWSC is transfering to the Air Force Materiel Command (AFMC) and will be known as the Space and Warning Systems Directorate (SWSD)..

[4] Formerly IBM Federal Systems Company

The SWSC, determined to apply new software technologies to reduce maintenance costs, had already been working to build up a megaprogramming capability; and the partnership with STARS was a natural way to accelerate the transition. Table 1 provides a summary of the progress to date on the project - in terms of the three megaprogramming technology thrusts being pursued by STARS.

	Original SWSC Posture	Accomplishments since Establishing STARS Partnership	Activities in Progress (as of 1/95)
Domain-Specific Reuse	• Strong architecture, based on RICC architectural infrastructure • Strong emphasis on Open Systems, commercial tools • Domain models underway • Commitment to Ada	• Demonstrated viability of architectural approach • Defined tailored specification standard based on Cleanroom, MIL-STD 498, and others • Completed object-based application models; specified SCAI system and first two SCAI releases • Developed and tested SCAI Release 1	• Developing SCAI Releases 2; specifying Release 3 • Refining product-line architectural framework • Continuing to develop domain models
Systematic Process	• Understanding of importance of process • SWSC Software Engineering Process Group (SEPG) established • Semi-formal process definition in selected areas • Corporate Information Management (CIM) IDEF model underway	• Instituted formal approach to process definition, based on STARS/SEI collaboration • Created a product-line process architecture • Integrated OO, Cleanroom, and the Ada Process Model methods • Formally defined processes for Application Engineering (AE) • Launched a major metrics initiative • Automated staff hour and defect metrics collection	• Nearing completion of formal definition of CM process • Beginning formal definition of Domain Engineering process • Working on second round of AE specification process • Using automated process modeling and enactment support for SCAI Release 2 and 3
Automated Support	• Commitment to Rational Ada support product-line • Commitment to Universal Network Architecture Services (UNAS), and Reusable Integrated Command Center (RICC) for Architectural Infrastructure	• Integrated a state-of-the-art open systems SEE: IBM and Sun platforms, Rational and TRW toolsets • Installed advanced process support toolset; encoded and began automated enactment of SCAI Release 2 and 3 processes • Instituted automated tracking capability for problems, action items, etc. • Began use of Rational SoDA for automated document production	• Enhancing the functionality and integration of the process tools • Applying Amadeus to automate collection of SEE usage metrics • Extending process automation across geographical locations

Table 1. Air Force Demonstration Project - Megaprogramming Progress and Status

2.3 DEMONSTRATION PROJECT SEE

The Demonstration Project SEE is composed of approximately 50 workstations connected to 3 server-class machines. The network is distributed across two geographical sites (over a high-speed link) and is about evenly divided between Sun and IBM Unix-based platforms.

The SEE is populated with a set of tools to support the desired end-user functionality, as depicted in Figure 1. The SWSC has committed to the use of Ada for the

application product-line, and has selected the Rational toolset (Apex, Verdix, RCI, Ada Analyzer, SoDA, and Rose) as a key part of the SEE. In addition, because the application architectural strategy is based on a TRW-originated architectural infrastructure, the SWSC has adopted the corresponding toolset (SALE, RICC Tools) as another key part of the SEE. The diagram also discloses the organization's emphasis on process technology (the Process Modeling, Project Management, Process Enactment, and Metrics functionality clusters).

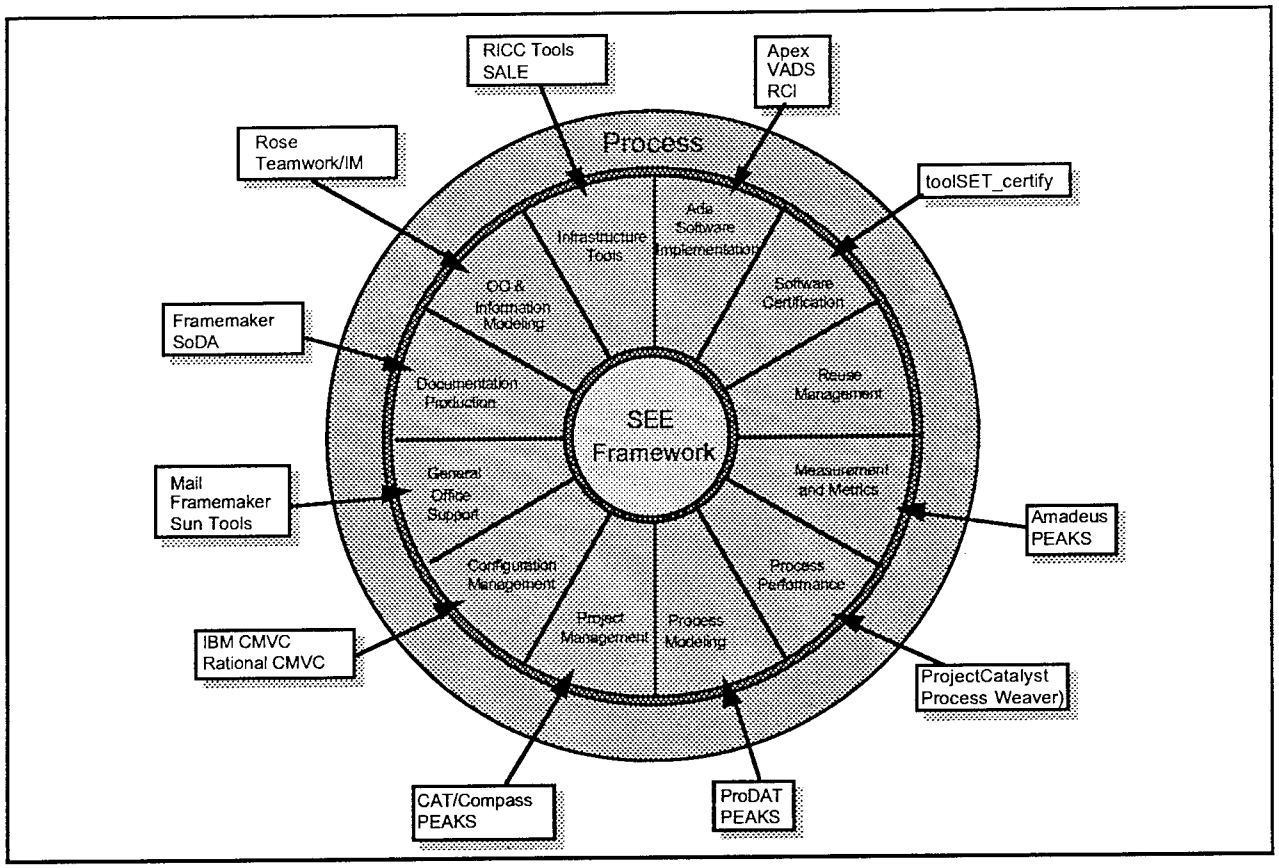


Figure 1. Demonstration Project SEE: Tool Functionality Groups

Table 2 identifies the supplier of each tool shown in Figure 1. As shown, the SWSC has selected Rational and TRW as major toolset providers for the Demonstration Project. Also shown in the table are four tools that have been developed with STARS support.

Type	Tool Name	Vendor/Developer	Purpose
Rational Ada Development Toolset			
	Apex	Rational	Code creation and testing
	RCI	Rational	Interfaces Apex with other Ada compilers
	Rose	Rational	Object oriented analysis and design
	SoDA	Rational	Automated document generation
	VADS	Rational	Verdix compiler
TRW Architectural Infrastructure Support Toolset			
	RICC Tools	TRW	Application display, message, and database definition
	SALE	TRW	Application network definition (used with UNAS)
	UNAS (Universal Network Architecture System)	TRW	Application network manager
STARS-Supported Tools			
	Amadeus	Amadeus Software Research, Inc.	Metrics repository and analysis
	PEAKS	Cedar Creek Process Engineering (ccPE)	Process modeling, planning, and plan simulation
	ProjectCatalyst	Software Engineering Technology (SET)	Low-level process definition and process execution
	toolSET_certify	SET	Cleanroom certification testing support
Other Tools			
	CAT/Compass	Robbins-Gioia	Project management
	CMVC	IBM	Configuration management and tracking system
	FrameMaker	Frame Technology, Inc.	Documentation and publication
	ProDAT	Embedded Computer Resource Support Improvement Program (ESIP), managed by Sacramento/ALC	IDEF-oriented process definition
	Process Weaver	Cap Gemini America	Process workflow manager
	SunTools	Sun	General office support
	Teamwork/IM	Cadre Technologies	Information modeling

Table 2. Demonstration Project SEE Tool Suppliers

The integration of the SEE tools is accomplished through a combination of techniques and mechanisms. Control integration (tool invocation and communication) is provided by

- IBM AIX SDE WorkBench/6000 broadcast messaging service, the Process Weaver message service, operating system process services, and TCP/IP sockets. The WorkBench and operating system process control provided local service within a user's machine session. Process Weaver and TCP/IP provided service between users and between machines.
- Data integration is provided by the Oracle Relational Database Management System (RDBMS) and the operating system file system.
- Presentation and user interface integration is provided by the X Window System and the Motif window manager.
- Process integration is provided by the STARS-sponsored Process Support Environment toolset: PEAKS, ProjectCatalyst (in conjunction with Process Weaver).

3. PROCESS-BASED SEE INTEGRATION

This section elaborates the notion of encapsulation layers, discusses their relevance to SEE integration, and offers a generic model of the Process Support Environment (PSE) encapsulation layer.

3.1 ENCAPSULATION LAYERING AND SEE INTEGRATION

The Importance of Encapsulation Layers in Software System Architecture in General

The importance of encapsulation layers is emphasized in [SchMel92][5]. To illustrate the notion of layering, consider a system for with a human-machine interface (HMI) layer which encapsulates the specifics of the operator's physical interface with the system (i.e., the display and console interaction). This example can be used to identify some of the hallmarks of encapsulation layers:

- The essential requirements for the layer are the rules about the system's usage paradigm plus the application programming interface needed by the system's higher levels
- The layer is expected to provide an implementation that binds to lower-level details - in this case, device and operating system characteristics

The power of such encapsulation stems from these characteristics:

- It allows reasoning about the capabilities of the layer with minimal concern about other aspects of the system

[5] Schlaer and Mellor use the term "domain" to refer to what we are calling "encapsulation layer".

- It allows maximum efficiency in applying experts in the layer's domain (in this case display interfaces)
- It facilitates replacing the layer's implementation (e.g. using new technologies) with minimal impact to other parts of the system

Architectural Layering in The Demonstration Project Application

On the Air Force/STARS Demonstration Project, architectural layering is viewed as essential to laying the foundation for a megaprogramming product line. Figure 2 provides a high-level depiction of the architecture being used for the SCAI application. Note that three major encapsulation layers provide the foundation for all applications in the future product-line. The lower two layers are off-the-shelf, and the third layer (architectural infrastructure) provides product-line specific encapsulations for such services as User Interface, Data Management, and Message Handling. Interested readers are invited to refer to a more detailed discussion of the SCAI architecture found in [Bristow95] and [Bulat95].

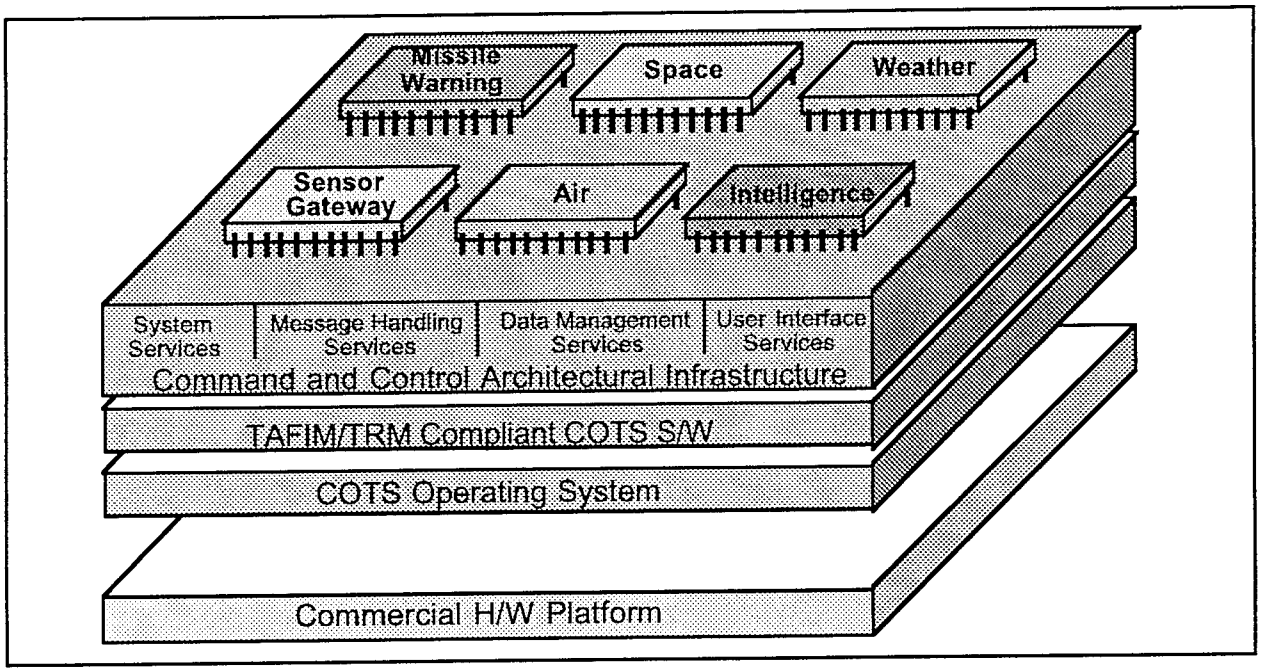


Figure 2. Layered Architecture for Demonstration Project Application

Architectural Layering Applied to the SEE

We now move on to discuss how the above general notions about encapsulation layers apply to SEE integration. Table 3 is taken from the current draft of the Air Force/STARS Demonstration Project Experience Report [DemExp95]. It analyzes several architectural characteristics of the Demo Project SEE. Perhaps the most important characteristic is the layered architecture: as discussed above, implementation of each layer can proceed with minimal concern for the specifics of the implementation of the other layers.

Major Architectural Characteristics	Example Sub-Categories	SCAI SEE Domain Implementation Examples
• Layered architecture	• Common underlying operating system environment • Process support environment • Major functionality encapsulations	• Unix, TCP/IP, NFS, X • STARS PSE toolset • Rational's integration family of Ada support tools (centered around Apex) • TRW's integration family of code generation tools supporting the architectural infrastructure (UNAS, RICC)
• Common user interface	• Common window-handling • Common window behavior look and feel	• Motif • Open Interface (from Neuron Data), Display Builder (from TRW)
• Common program interface mechanisms	• Low-level API services • High-level data repository services	• Broadcast Message System (part of HP's SoftBench) • TCP/IP Sockets (for guaranteed message delivery) • COTS DBMSs (Oracle, Sybase)
• Component reuse	• COTS tools	• Various

Table 3. Architecture Characteristics for the SEE Domain

Figure 3 provides a view of SEE architecture in terms of some typical encapsulation layers, including the Process Support Environment layer which is the focus of this paper. The arrows show dependence of one layer on another (i.e., we are not showing data flow or control flow here).

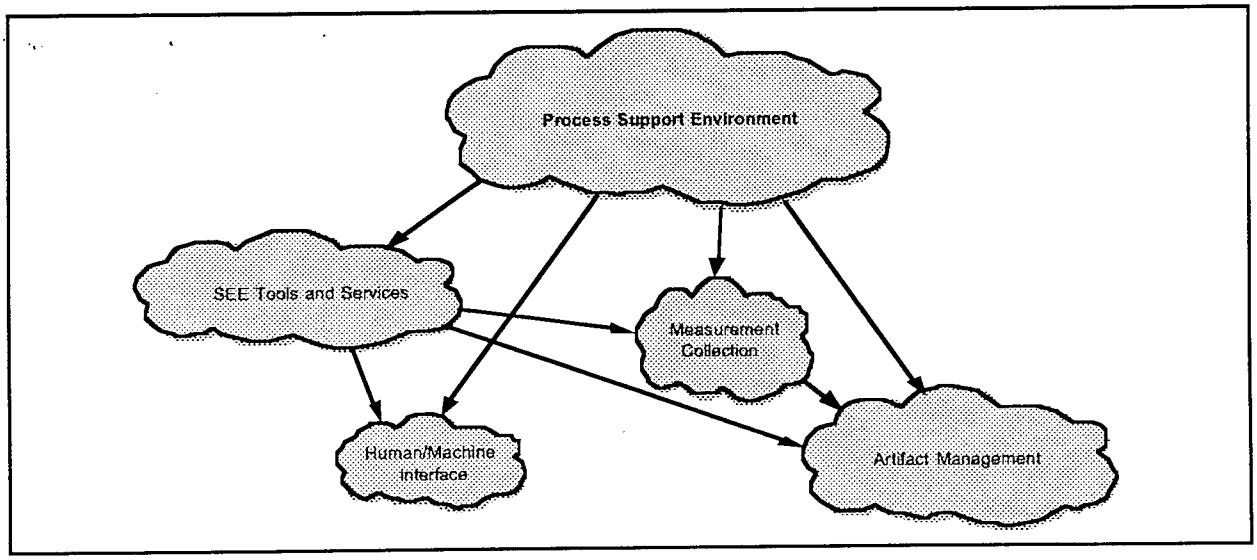


Figure 3. SEE Encapsulation Layers

Encapsulation Layers: A Fundamental SEE Integration Strategy

Just as with other software systems, a SEE integrated using well-engineered encapsulation layers will exhibit more effective functionality and will be more maintainable and evolvable, since:

- Integration *within* encapsulation layers can take advantage of the knowledge and expertise of a relatively small number of experts, where experience has shown that tight integration is the most successful.
- Less integration will be needed *between* distinct encapsulation areas (simplifying maintenance), and can be restricted to standardized interfaces (facilitating replacing a layer with an improved alternative).

Implementing SEE Encapsulation Layers using SEE Integration Mechanisms

It is important to distinguish "encapsulation layers" from another term that is commonly used in discussions of SEE integration: "integration mechanisms". An encapsulation layer is an encapsulation of functionality, while an integration mechanism is a means of joining components for the purpose of combining functionality. The components within a layer may be joined together via integration mechanisms - and, further, the layer as a whole may be joined with other layers via the same mechanisms.

To implement these encapsulation layers and to interconnect them, integration mechanisms are used. Here are the most commonly cited integration mechanisms[6], with brief examples for how each is used:

- ***Control integration mechanisms*** permit the invocation of a service provided by another component (perhaps including the launching of an entire application). Such mechanisms include direct calls, sockets, mailboxes, and broadcast messaging. Examples of the facilities providing the latter type of services are Hewlett-Packard's SoftBench[7] Broadcast Message Service (BMS) and Sun's ToolTalk[8].
- ***Presentation integration mechanisms*** permit the development of application wrappers that provide users with a uniform interface for the invocation of SEE applications and services. An excellent example of a presentation integration service is the one found in HP SoftBench.
- ***Data integration mechanisms*** permit application developers to share common data objects across applications, where each application developer uses a database or object manager, making the API for these applications known to other application developers. Data integration between heterogeneous software application vendors has had limited success.

---

[6] See, e.g, the August 1993 NIST/ECMA reference model for SEE frameworks [NIST93]

[7] HP SoftBench is a registered trademark of the Hewlett-Packard Corporation.

[8] ToolTalk is a registered trademark of Sun Microsystems.

- ***Process integration mechanisms*** support the coordinated use of the SEE's other integration mechanisms (control integration, data integration and presentation integration mechanisms) to support the execution of work steps within an automated process sequence (or task).

  Figure 4 illustrates the use of a process integration mechanism to bind a conceptual representation of a process segment, shown at the top of the figure, to specific SEE services and data - using control, data and presentation integration services.

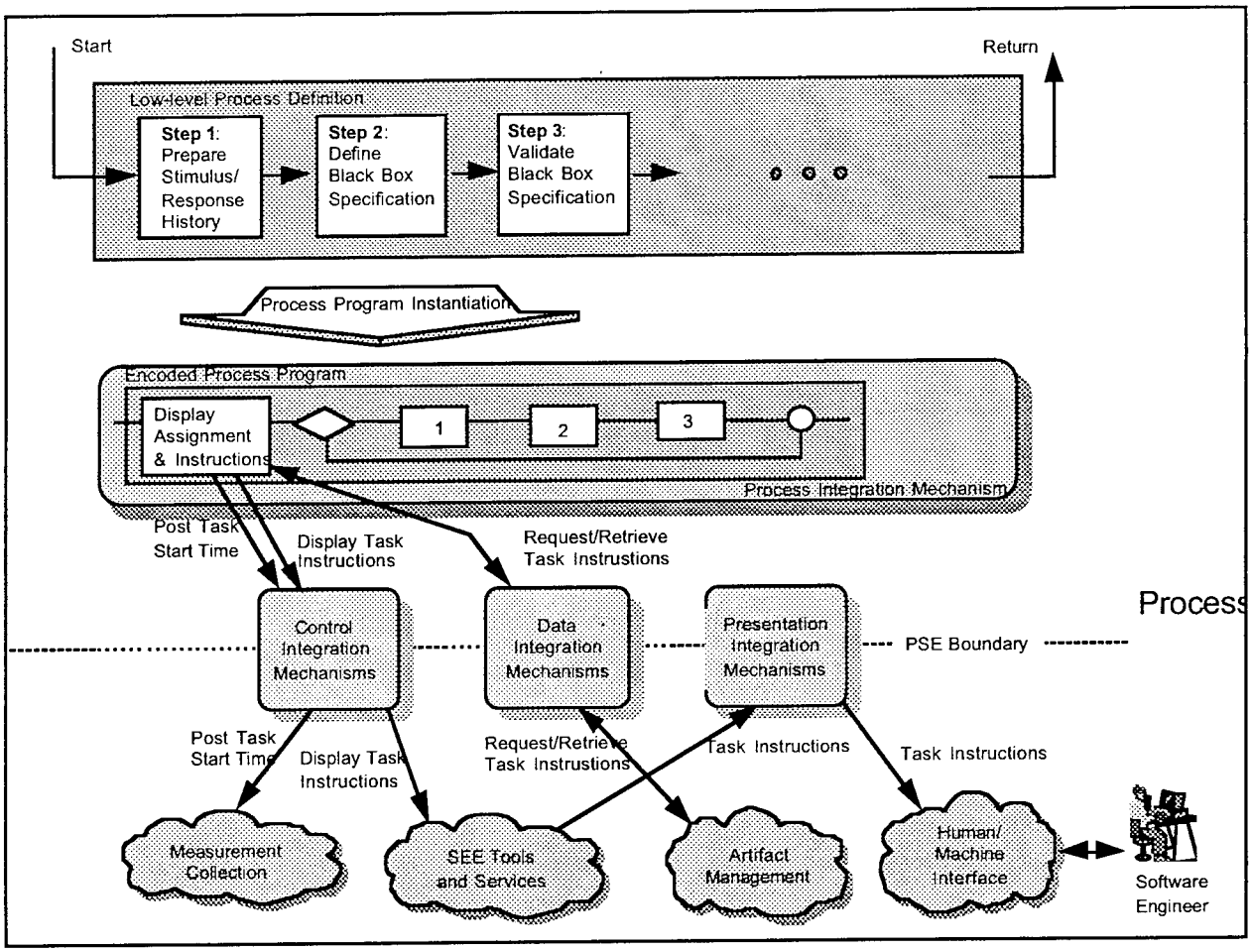


Figure 4. Supporting a Task via Process Integration (1 of 2)

The conceptual view is translated into a process program view, shown in the middle of the figure, in which process engineers have arranged for a new step to be executed: "Display Assignment of Instructions". This new step, which only makes sense when the process segment is supported by the PSE, posts the task start time to a measurement database and causes detailed process enactment description information to be automatically displayed to the practitioner.

This process program is executed by a "workflow engine", which uses an encoded representation of process sequencing to progress through

the steps. Associated with such an engine are process programming facilities that allow process engineers to bind each step to SEE artifacts and tools. The workflow engine thus provides the facilities to implement process integration.

As shown in Figure 4, the execution of the first step involves the use of the three other integration mechanisms to record the prescribed measurement data and to display the process instruction data.

Figure 5 is a continuation of this same example, showing how a later step, "Define Black Box Specification" is supported.

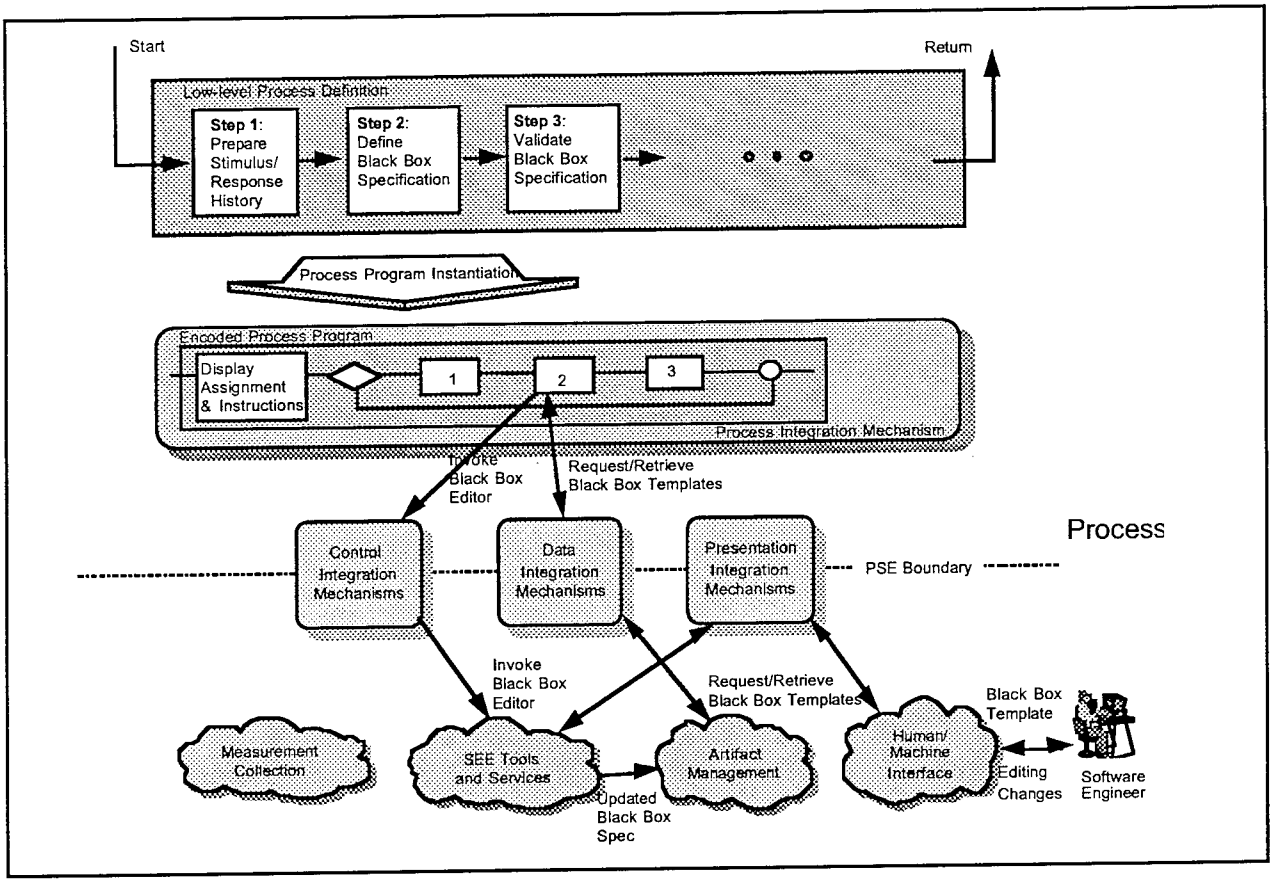


Figure 5. Supporting a Task via Process Integration (2 of 2)

This step is implemented by using a Control Integration Mechanism to invoke an editor, and a Data Integration Mechanism to retrieve the project's standard template for black boxes.

Examples of commercial products that provide process integration services - also referred to as workflow automation services - are HP Synervision[9], Process Weaver[10], FlowMark[11] and InConcert[12].

[9]HP Synervision is a registered trademark of the Hewlett-Packard Corporation.

[10]Process Weaver is a registered trademark of Cap Gemini Innovation.

Please refer to Section 4.3, Lessons Learned, starting on page 23, for a discussion of some of our experiences in applying SEE integration mechanisms.

3.2 THE PROCESS SUPPORT ENVIRONMENT ENCAPSULATION LAYER

The foregoing discussion has developed the importance of encapsulation layers to SEE integration. The remainder of this paper focuses on one particular encapsulation layer: the Process Support Environment (PSE) layer. The authors' contention is that this layer is one of the most vital SEE integration priorities, because it directly joins the organization's process to the SEE that supports it. As will be shown, use of a PSE can not only tie the SEE's functionality together for the end-user, it can also keep the process alive - and improving - by making it part of every user's routine use of the SEE. This dual integration strategy is illustrated in Figure 6.

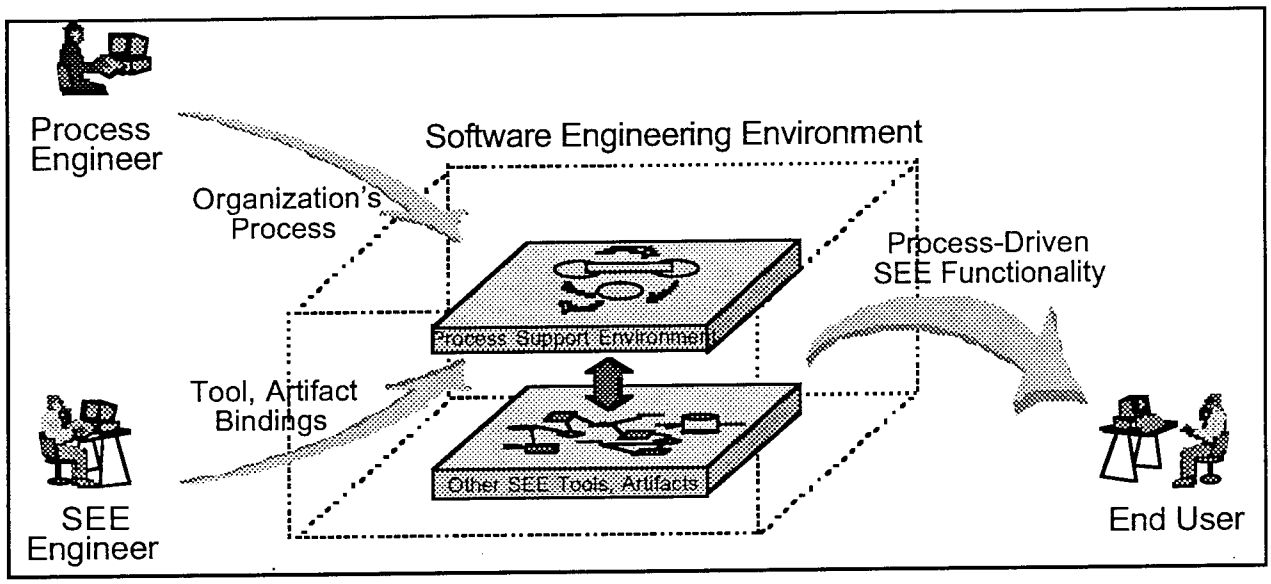


Figure 6. Process Support Environment Encapsulation Layer

Conceptually, instantiation of the process integration layer proceeds via a "lamination" procedure: the process and metrics requirements are added to the top, and the tool interface provisions are added to the bottom. The result is an encapsulated SEE integration layer that joins process to SEE.

A list of criteria for recognizing encapsulation layers was provided on page 7. As shown by the following analysis, the PSE encapsulation layer seems to fit the mold:

- The essential requirements are provided by the process, since the objective of the SEE is to support the organization in carrying out its process

[11] FlowMark is a registered trademark of the IBM Corporation.

[12] InConcert is a registered trademark of the Xerox Corporation.

- The "API" with the system's "higher levels" are the user interfaces:
  - For process engineers, to define and model the process
  - For managers, to plan a project based on the process
  - For practitioners, to carry out their work in the context of the process
  - For all users (including the above), to assess how well the process is working and to pursue high-payoff avenues for process improvement
- The organization can reason about the process integration layer with minimal concern for the specifics of the rest of the SEE
- The implementation of the layer is performed by process technology experts that are intimately familiar with state-of-the-art process thinking, available process tools, and SEE integration techniques
- Customization provisions allow organizations to modify other aspects of the SEE - such as changing compilers or text processing toolsets - with minimal perturbation to the layer or its users.

3.3 A Look Inside the Process Support Environment

This subsection provides a model of how such a Process Support Environment (PSE) might be constructed. The model will provide the basis for our discussion of the Demonstration Project PSE, and it will also assist in comparing and contrasting our approach with that of others.

3.3.1 PSE Requirements

Our view of the PSE's structure is motivated by the following general requirements.

The PSE should:

- Provide process automation options ranging from guidance to complete automation; the degree of process automation is always the organization's prerogative.
- Allow incremental process definition, implementation, and improvement.
- Support process definition via graphical modeling as well as conventional documentation. Preferably, the modeling syntax should be geared to describing process and the semantics should permit consistency checking.
- Support "process-driven project management". By this we mean:

- Allow project plans to be developed and evolved in the context of the organization's process. Allow project managers to identify the process basis for planned project activities.
- Allow management control to be tied to the process-driven plan. Managers and task leads should have the ability to discern the process context of the activities they are coordinating.
- Allow status monitoring to be automatically tied to process execution. Managers and task leads should be assured that milestones are reached via the defined process - including the successful completion of any predefined validations.

We regard management support capabilities to be part of the PSE because of the intimate relationship between process and management.

- Support automated assistance of process execution, providing process guidance, facilitating artifact navigation and tool usage, and enforcing conformance to process-defined standards in as unobtrusive a fashion as possible.
- Support measurement capture during the course of process execution, automatically where possible.
- Support metrics analysis, based on the above process-driven measurements as well as other means (e.g., analysis of source code), as a key ingredient of process improvement and technology transition.

The current Demonstration Project PSE partially satisfies all of the above requirements.

3.3.2 PSE Structure

Figure 7 provides a abstract view of the PSE (the shaded regions of the figure), showing its internal structure and its relationship to the rest of the SEE. The figure identifies several distinct ***capability sets*** within the PSE:

- Process modeling
- Project planning, monitoring and control
- Process performance management
- Task execution
- Process improvement analysis

Figure 7 also shows the PSE layer using services in the other encapsulation layers shown on Figure 3 on page 9, namely:

- SEE tools and services (e.g., compilers, object-oriented modeling tools, etc.)

- Measurement collection services for posting, storing and retrieving measurement data
- Artifact management services for accessing and storing artifacts and information about artifacts - including configuration management.
- Human/machine interface services (not explicitly shown in the figure).

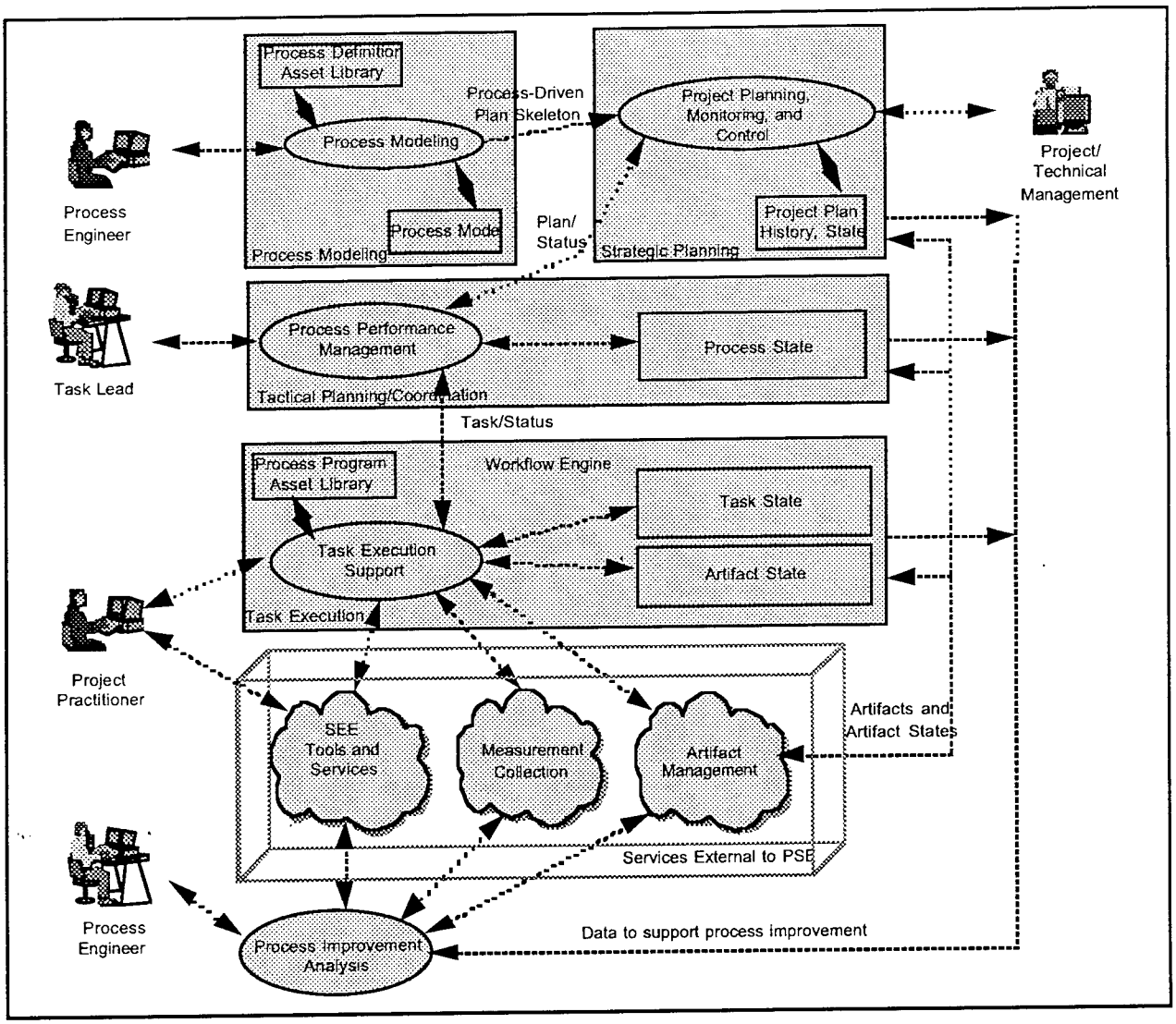


Figure 7. Process Support Environment: Structure and Interfaces

3.3.3 Description of PSE Capability Sets

Process Modeling (PM) Capability Set

This set provides process engineers with the tools necessary to define the organization's process, comprising:

- An architecture for the process, describing all of its components and their relationships
- The flow of artifacts within a process and its components (data flow)
- A constrained process activity network suitable for supporting project planning
- The control flow and control structure of activities within a process and its components (process/process component control structure)
- The composition and description of the artifacts involved in the process
- The quality characteristics established for each artifact the project is to produce and the completion criteria each artifact must satisfy
- The resource types required to support each identified activity
- A description of each process component, such that each process component has sufficient information available for its user to manually perform it.

Process specifications that are defined and tailored using the PSE's process modeling capabilities are maintained in a process asset library for use by current and future projects.

The process specifications prepared and adopted to support a project, become a key ingredient in the planning, design and implementation of process programs. A process program is a program designed to provide assistance to SEE users in following a defined process, as they perform their project work.

Project Planning, Monitoring and Control (PPMC) Capability Set

This set supports project and technical managers working with process engineers to initially plan software development projects. Further, once these projects are initiated, this capability set provides support to project and technical managers who monitor and control the project.

The PPMC Capability Set provides support for planning software development projects by providing the following capabilities:

- Automatic project plan instantiation from process specifications that provide the constrained activity flow for the project
- Analysis of proposed project plans for cost and schedule reasonableness.

The PPMC Capability Set provides support for process component delegation to technical management and their task leaders to address how they plan to satisfy the objectives of a delegated process.

Once a project is initiated and processes have been delegated for detailed planning and management, the PPMC Capability Set provides support for monitoring and controlling the project though the following capabilities:

- Automatic posting of project status and events for weekly, bi-monthly and monthly project status report generation and analysis
- Programmable project exception reporting on selected project events, e.g. schedule anomalies, cost anomalies, resource anomalies, etc.
- Support for project replanning, upon process conditions requiring the project schedule to be revised, e.g. review failures, earned value problems, etc.

Project activity and process state information is kept in a persistent object store for use in supporting report generation and automated event recognition.

Process Performance Management (PPM) Capability Set

This set supports technical managers and task leaders who plan and manage the detailed tasking necessary to address the specific requirements of a delegated process. As processes are not intended to dictate how technical work will be performed, technical managers and task leaders must plan the tasks they feel are necessary to achieve expected results and to coordinate these activities with their technical team.

The PPM Capability Set provides a process/task planning and management spreadsheet for use in defining and coordinating lower level processes and the tasks necessary to satisfy them. This spreadsheet provides the following capabilities:

- Low level process definition (processes below those of the ones defined in the project plan)
- Task planning to support the requirements of a delegated process
- Process delegation and tracking
- Task assignment, dispatching and tracking.

The process/task planning and management spreadsheet displays an overview of every process task, and as work proceeds and project milestones and process events occur, the spreadsheet is updated to reflect these changes. Detailed information is summarized for automatic reporting to support the PPMC Capability Set.

Project task state information is kept in a persistent object store to support the summarization of detailed data for use in supporting report generation and automated event recognition. Project artifact state information is also kept in a persistent object store to provide information for the PPM Capability Set about artifacts that exist and their usage, completeness and version status.

It is at the PPM Capability Set level that requirements for designing and implementing process programs must be made. Process programs may either be custom crafted for each process in the project's process architecture, or generic process programs may be prepared to support activity classes. The PPM Capability is responsible for dispatching tasks for project personnel to perform. This task dispatching in-

volves the invocation of a process program to support project personnel in performing their assigned task.

Task Execution Support (TE) Capability Set

This set supports project personnel in performing their work tasks, and provides task status information to the PPM Capability Set to support its mission of task tracking. It includes a workflow engine programs (process integration mechanism) for executing low-level process programs. The engine uses other SEE integration mechanisms (e.g., control, data, and presentation mechanisms) to interface with services external to the PSE layer. For example, a process program may automatically invoke a SEE tool, and a few steps later, request a measurement engine agent to post a process measurement.

The TE Capability Set may be implemented according to one or more automation paradigms:

- **Passive support** for project personnel, through the use of electronic mail and a hypertext description of the process currently being performed.
- **Complete automation**, where shell scripts (or canned procedures) are launched based on a SEE or process event.
- **Proactive work flow management,** involving both a human and the SEE. In this instance, work is assisted by an engine that provides convenient access to the appropriate tools and data - and provides guidance to support the practitioner's performance of task steps.

Since the latter automation paradigm is the most general capability (a coordinated mix of human and SEE activities) we will assume PSE capabilities include a proactive work flow execution support, employing a work flow engine, such as Process Weaver or InConcert.

When a task is dispatched by the PPM Capability, a process program to support that task begins its execution. Each process program includes a set of work steps that a task performer must accomplish. These work steps are made available to the task performer, based on the process program's control structure. Thus the process program, using the work flow engine, manages a dispatched task through to its completion. Process programs may be instrumented to provide automated measurements and validation steps:

- Task start and completion times can be automatically recorded, as well as task accumulated elapsed and environment usage time. This instrumentation is typically done by interfacing with a measurement engine, where measurement collection agents are invoked to collect and store the require measurement.
- Product reviews can be facilitated, by prompting with prescribed questions about the quality of an artifact that an engineer produced.

To increase the level of support, TE can provide automated launching of tools against the key artifacts being consumed and produced by the task. To accommodate this, a binding strategy must be implemented within the PSE, as indicated in Figure 8. Such binding can allow TE to simplify the practitioner's work by eliminating the need to navigate to the proper artifact and invoke the appropriate tool for it.

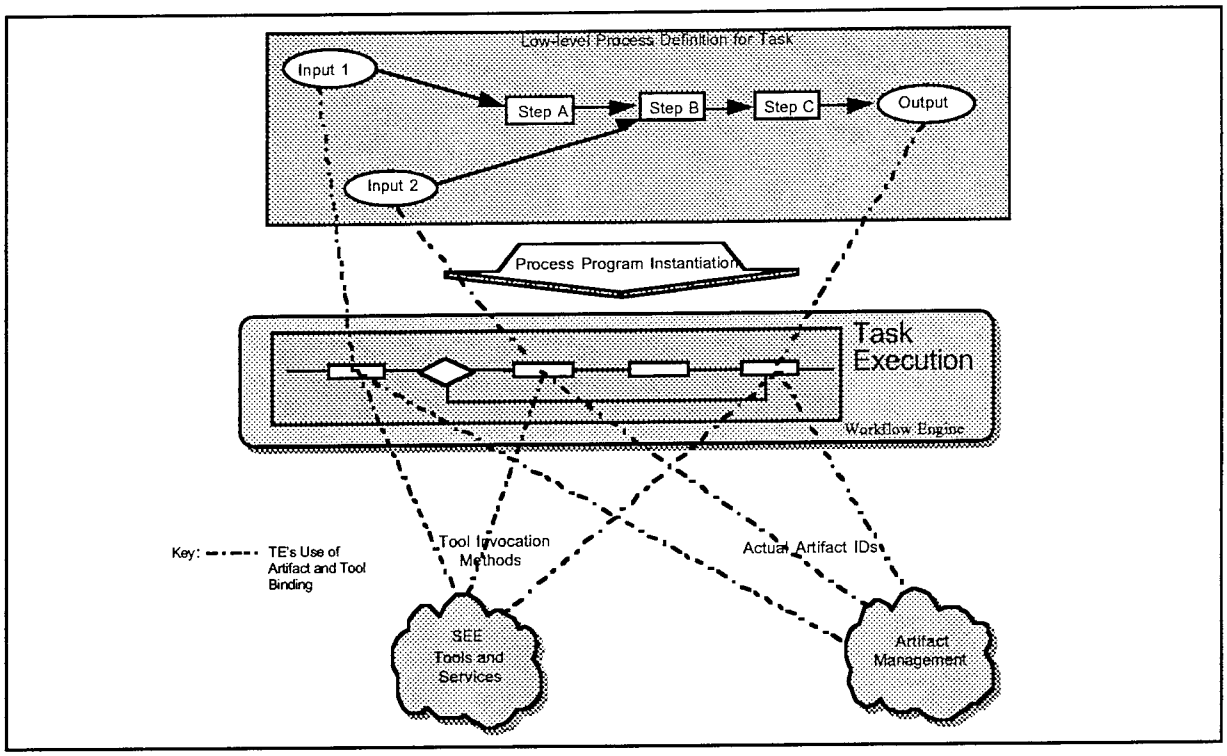


Figure 8. Binding Process Programs to SEE Artifacts and Tools

Global and local task state information is maintained in a persistent object store for the work flow engine to support task execution, intertask synchronization, task redirection, or task suspension. TE also maintains artifact state information to support task execution, intertask synchronization and task suspension.

Process Improvement Analysis (PIA) Capability Set

This set provides process engineers with access to the tools and resources to support both quantitative and qualitative analysis of process performance data and product quality data. Process engineers employ many techniques to support process improvement from reviewing the comments of process practitioners (project personnel who follow processes) to statistical analysis of data collected by the PSE Capabilities and the Measurement Collection Capabilities. Process engineers may employ statistical analysis tools to analyze process performance and project quality data to analyze performance times and defect ratios against existing norms. Further, using measurement data collected during process execution, metrics may be computed to support process improvement analyses. Typical PIA Capabilities include:

- Statistical analysis support for trend analysis and curve fitting

- Control chart preparation and analysis
- Problem cause and effect analysis.

3.4 Related Work

We have provided an appendix to the paper, "History of PSE Evolution", starting on page 34, to provide additional background to interested readers. The appendix discusses three antecedent PSEs developed under the Loral Federal Systems STARS contract, as well as a contrasting PSE under development on the Arcadia project. The PSEs are all analyzed in the context of the model of PSE structure and interfaces illustrated in Figure 7 on page 16.

4. The Air Force/STARS PSE: Experience To Date

4.1 PSE Description

Figure 9 illustrates the current Demonstration Project PSE. The diagram shows how the major PSE model components work together to support the process definition/planning/performance/improvement cycle.

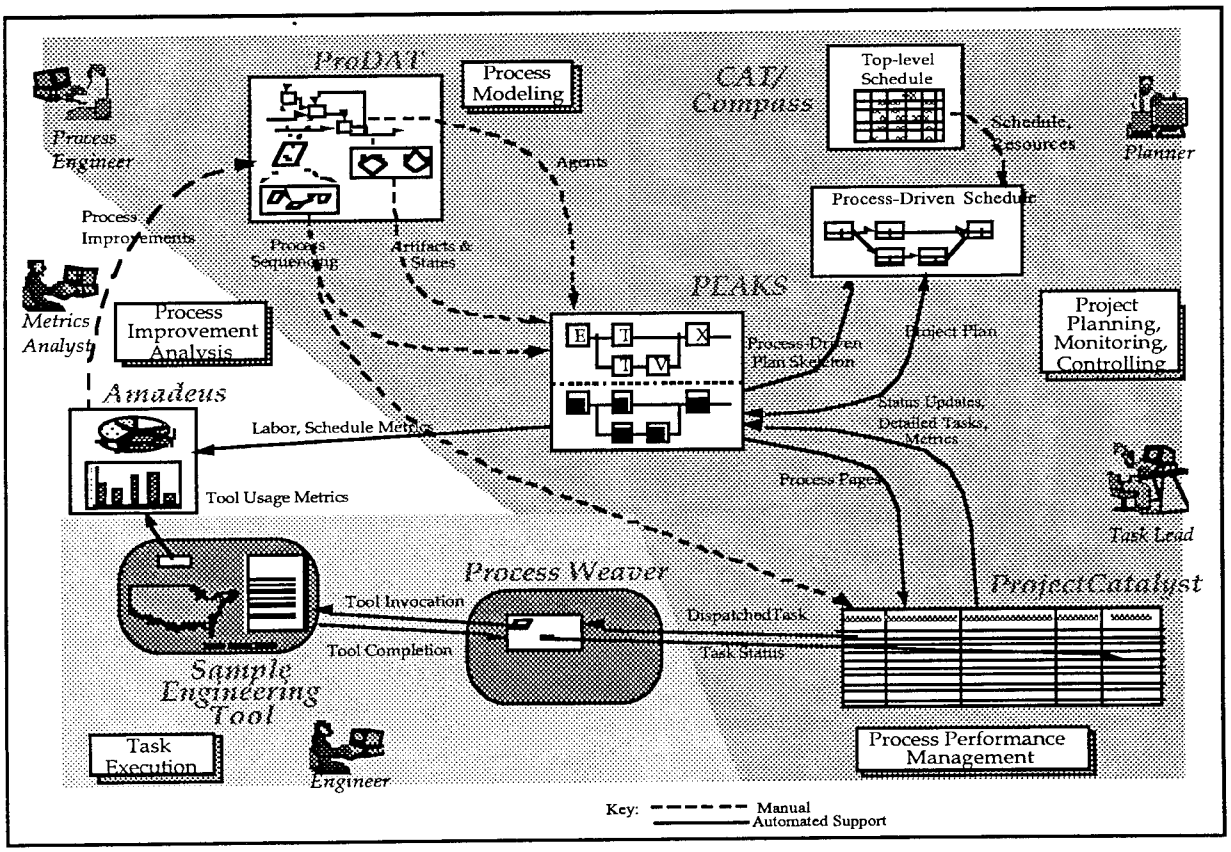


Figure 9. An Operational View of The Demonstration Project PSE

Process definition and modeling are supported by ProDAT, for activity modeling, and by the STARS-sponsored Process Engineering and Kernel System (PEAKS[13]), for work flow. These tools are being used by the Demonstration Project team to capture and analyze process in a form consistent with the STARS/SEI Process Definition Information Organizer Templates.

Process-driven project management is supported by PEAKS, which builds project plans directly from the process and allows management to apply resources and schedule constraints - as well as to simulate the impact of process modifications and quality failure probabilities.

Process enactment is supported by the STARS-sponsored ProjectCatalyst, and Cap Gemini's Process Weaver. ProjectCatalyst receives enactment specifics from PEAKS and allows task leads to manage the team's activities to follow the process-driven plan. Based on the project's customization information, ProjectCatalyst automatically instantiates Process Weaver process used by the practitioners to launch the appropriate tools for creating and manipulating the desired artifacts. As work actions are taken (starts and completions), ProjectCatalyst reports task completion date and effort to PEAKS, so that management can always obtain an accurate data concerning task and project status.

Measurement collection and metric computation is supported by both PEAKS and Amadeus. PEAKS automatically captures measurement data about the work as it progresses, thanks to the ProjectCatalyst reporting discussed above, and it provides relational calculus query capability to compute statistics. PEAKS also gathers pre-specified quality metrics (such as code complexity), which can be interpreted in the context of one or more quality frameworks adopted by the project (such as the RADC Quality Framework). Amadeus provides a repository for capturing arbitrary metrics data, together with a set of analysis and reporting aids.

4.2 History and Current Status of the PSE

The STARS components of the Demonstration Project PSE (PEAKS and ProjectCatalyst) were introduced in late 1993 and early 1994. A major pilot was conducted in March of 1994, and based on the results of the pilot, the project began use of these tools for the SCAI Release 2 Specification activity. Both the pilot and the operational use of these tools provided substantial practical experience on both the specific implementation of the tools and the process-driven paradigm supported by them. The main problems that emerged were:

- The toolset was not sufficiently stable to support production work;
- PEAKS and ProjectCatalyst were not fully integrated, resulting in redundant manual work to translate PEAKS models to ProjectCatalyst process performance pages;

[13] PEAKS is the commercial name chosen by ccPE for the STARS-supported process modeling facility formerly known as Software Process Management System (SPMS). Since the name change is quite recent, most STARS publications, including [DemExp95] use the SPMS name.

- ProjectCatalyst tool and database administration was excessively complicated; and
- The ProjectCatalyst implementation of the "State Data Repository" (SDR), originally designed to support text files, was not sufficient for the project's heterogeneous artifact types or its configuration management requirements.

These problems were largely due to the unprecedented nature of the usage paradigm being supported by the tools, and a resulting under estimation in the complexity of the required software.

Despite the problems, the project decided that there was sufficient promise in the toolset to plan a major upgrade. The upgrade, targeting all but the last of the above problems, was delivered on schedule in November, 1994, and is now being used for the SCAI Release 3 Application Engineering Specification activity. The fourth problem - the need to provide a more mature artifact management capability - remains a future objective.

During this same period, an affiliated team, funded by the Air Force's Sacramento/ALC EISE program at Sacramento, California, was conducting a feasibility study on the development of a tool to support both $IDEF_0$ activity-based process modeling and $IDEF_{1X}$ data modeling. This work yielded the ProDAT tool, based on the VSF[14] engine. After conducting a pilot of ProDAT in September, 1994, the project decided to use ProDAT in place of DesignIDEF and Framemaker, to reduce the complexity of assembling IDEF-based process descriptions, from a tool with a single database, as opposed to the more manual intensive approach that was being employed.

Figure 9, on page 21, depicts the PSE in use on the Demonstration Project as of this writing.

4.3 LESSONS LEARNED

The lessons presented in this section are derived from roughly five years of experience in studying process and process automation technology, and piloting a succession of Process Support Environments (PSEs).

- ***Building a Process Support Environment is an ambitious undertaking.***

  We underestimated the difficulty in building and transitioning the STARS portion of the Demonstration Project PSE (PEAKS and ProjectCatalyst/Process Weaver).

  In retrospect, there are several reasons for the underestimation:

  – Despite prior experience with PSE antecedents (refer to the Appendix to this paper), the usage paradigms and ingredient technologies were still largely unprecedented.

[14] Virtual Software Factory (VSF), provided by VSF, Inc., is a tool specifically designed for developing graphically-oriented computer-aided software engineering (CASE) tools.

- Prior experience (e.g., with the Cleanroom Engineering Process Assistant[15]) was in simpler usage environments and with a smaller user population. The Demonstration Project SEE requirements were more complex than envisioned, due to such factors as the heterogeneous hardware and software environment and use of multiple SEE frameworks.

- Work prior to the identification of Loral's Demonstration Project partner was geared to the Cleanroom software engineering process - which had already been specified, tooled, and piloted with predecessor PSEs. Although the new organization decided to incorporate Cleanroom principles in their process, it turned out that most of the prior work had to be rethought.

  This factor relates to a larger project-level lesson learned cited in [DemExp95]: a project must work out its own product-line process - organizations will seldom be able to use an off-the-shelf process defined by another party.

- ***Attempting to make a large number of significant approach changes on a project can impede the progress made in any one area.***

  This is another example, cited in [DemExp95], where larger project-level issues affected work in the SEE area. The Demonstration Project as a whole had very ambitious technology objectives, resulting in simultaneous development of basic approaches on several fronts at once, including:

  - Process definition methodology,
  - Process content (numerous advanced processes are targeted by the SWSC, including domain engineering, product-line configuration management, metrics, etc.), and
  - Process automation.

  Adding to these approach issues were the engineering issues associated with building the SCAI application itself. Although the project has done remarkably well at synthesizing a promising approach in nearly all of these areas, each one of them suffered to some extent from unknowns and variables - as well as from competition for intellectual energy.

- ***Transitioning to a significantly new approach requires an incremental strategy.***

  Another lesson we re-learned was that of technology transfer techniques. When it comes to radical new ideas, such as process-driven

[15] Cleanroom Engineering Process Assistant; described in the Appendix on page 34.

development, baby step introduction of new technology, followed by successful technology use is far better than trying to introduce too much technology, too soon. Once this approach was taken, technology introduction and adoption was greatly facilitated. Performing pilot products to permit a customer to gain experience with a new technology is vital to the adoption process.

- ***There are additional SEE integration challenges to be addressed to improve the practicality of a PSE.***

  While we expect the PSE concept to pay rich rewards in the long-haul, there are some integration challenges ahead. We believe the following process integration lessons - and the experience upon which they are based - represent one of the most significant results of the investment in process technology made by both STARS and the Demonstration Project organization.

  - ***Differences between platforms, sometimes quite subtle, can complicate integration.***

    For example, one of our integration problems was the control and data integration of PEAKS and CAT/Compass[16]. A message set was carefully planned to support data exchange between PEAKS and CAT/Compass. The testing of the individual message sets against the two tools was also successful. However, the control integration mechanisms tailored on one environment (SUN UNIX) exhibited different behavior on the other (IBM AIX). Problems in permitting full message exchange between these two tools were never fully resolved.

  - ***Integrating a new tool as a means of gaining "off-the-shelf" technology is often more complicated than anticipated.***

    As an example, we sought to integrate a commercial project management package with PEAKS so that we could realize "process-driven planning" without implementing planning functionality in PEAKS. We soon found that we had to define a fairly complex interface specification, because the tools' perceptions of apparently similar data turned out to be quite dissimilar when examined in detail.

  - ***The "multiple database problem" is a pervasive integration issue.***

    First, if two related tools are left un-integrated, the user is forced to manually keep the respective databases in agreement. Since this can be a formidable challenge, one or the other database is likely

[16] CAT/Compass is a management planning package from Robbins-Gioia. CAT and CAT/Compass are registered trademarks of Robbins-Gioia, Inc. of Alexandria, Virginia

to fall behind, diminishing or nullifying the value of the associated tool. Second, when integrating two tools with overlapping functionality, each side must understand the other's database so that they can formulate a strategy for keeping each other's data in sync. Third, given that integrating the two tools is feasible, careful design is needed to be sure no messages are lost.

- ***Integrating separately developed process support tools can involve modeling paradigm issues as well as SEE integration issues.***

  In view of the prior lesson, the Demonstration Project would ultimately like to smoothly integrate all of the PSE capabilities depicted in Figure 9 on page 21. One of the key issues in achieving this integration is the differing modeling points of view taken by the tools.

  The Air Force is mandated to use the $IDEF_0$ notation for activity modeling, and the SCAI project opted at the outset to use MetaSoft's DesignIDEF tool for developing its process framework model. Unfortunately, despite its strength for understanding activity relationships, $IDEF_0$ is not sufficient for specifying enactable processes, because it is not capable of specifying constrained activity flow and control.

  PEAKS, part of the STARS toolset, does model constrained activity flow and control, but does not have an $IDEF_0$ view. Thus, the objective of integrating the two databases was thwarted not only by tool issues, but more importantly, modeling paradigm issues.

  Although this remains a long-range issue, two improvements are already underway. The first is the conceptual integration of the $IDEF_0$ view into a consistent enactable process specification in the STARS/SEI Process Definition Information Organizer Templates, worked out in cooperation with the Demonstration Project [Ett3-94].

  The second is the transition to a new prototype IDEF-based modeling tool called ProDAT. ProDAT first integrates $IDEF_0$ activity modeling with $IDEF_{1X}$ information modeling. It then adds a new artifact state transition modeling view and uses the new view to generate process sequencing.

  Assuming the new ProDAT paradigm proves productive for the Demonstration Project, it may be possible to pursue tool integration between ProDAT and PEAKS.

- ***Message-based control integration services require careful design to assure key data is not lost.***

For control integration among PSE components, we chose to use broadcast message server capabilities provided in IBM AIX SDE Workbench. The BMS metaphor is that a sender places a message on a software bus and one or more listeners hear the message and take appropriate action.

A central precept of process integration is that information used to support process control decisions must not be lost. Implementing PSE integration using BMS proved to require more engineering than expected, because message delivery is not assured. Use of BMS for process control requires considerable attention to such protocol issues as assured start up of listeners, handshaking, and error recovery. Redesign of our integration has led to a much more robust interface, but the potential still exists for message loss.

There were other limitations in Workbench that surfaced during the SEE integration effort. For example, communication between tools was supported only within a single user session. We took advantage of Process Weaver's inter-user messaging capability to help offset this problem.

The IBM Workbench product is an implementation of the Hewlett-Packard Broadcast Message Server (BMS) technology. We believe many of the Workbench limitations were the result of the lagging implementation of BMS improvements.

The emerging standard for SEE integration messaging is found in Sun's ToolTalk. This technology is part of the emerging industry standard referred to as the Common Desktop Environment (CDE). All the major environment platform vendors (Sun, Hewlett-Packard, IBM, Digital, and others) have agreed to provide CDE compliant products with their platforms. Future releases of AIX should provide an implementation of BMS (through ToolTalk) that will ease most of the limitations that we encountered.

- ***Data integration is impeded by toolsets that provide monolithic artifact management functions.***

In today's market, most toolsets that provide large amounts of functionality often perform self-contained artifact management - from their own points of view. In part, this stems from the absence of agreed-upon standards for common artifact management. The resulting mismatches make it difficult to implement a coherent artifact management encapsulation layer (as shown in the conceptual model of the SEE in Figure 3, on page 9). On the Demonstration Project, ProjectCatalyst was delivered with a self-contained "State Data Repository" (SDR), which both stored artifacts and maintained artifact states from a

process point of view. Rational Apex, used for Ada code development, manages its own artifacts, greatly enhancing its performance due to its detailed understanding of Ada semantics. Apex also implemented a much more sophisticated Configuration Management strategy than the SDR. Finally, the project selected IBM CMVC for its general-purpose CM solution, since it provided problem tracking capability, which Apex lacked. The net is that there are three artifact management methods available on the SEE. Currently, the project has decided to avoid use of the SDR entirely, and it is working on a locally written integration between CMVC and Apex.

Ultimately, as discussed out in [Randall89], what is needed is a general-purpose artifact management repository for the SEE - together with a standardized API - so that projects could replace one repository implementation with another as better solutions appeared. This is an active area of R&D, and no agreed upon approach has emerged. In the meantime there seem to be only two solutions for toolset vendors: continue to pursue independent monolithic implementations, or define a minimum essential API for an abstract repository and attempt to adapt one or more commercially available repositories to that API.

- ***Process state management should ultimately be handled by a central SEE server.***

  Our experience has shown us that ideally, process state should be managed by a server that all process support tools may access, and selected tools may update. The concept of the process server was also independently arrived at by the Arcadia project [Heimbigner95]. Failing an effective process server, a single tool should ideally manage and provide process state data to all requesting tools.

- ***Generic instantiation techniques can greatly reduce the need for process programming.***

  The PSE evolution work leading to the LORAL STARS PSE toolset (described in appendix A) brought forth the idea of an adaptable process support environment layer with a defined set of APIs that could be easily interfaced to employ SEE integration mechanisms. In terms of the PSE model shown in Figure 7, on page 16, these innovations apply to the interface between the Tactical Planning/Coordination (TPC) and the Task Execution (TE) capability sets. In the case of the STARS toolset, TPC capabilities are provided by ProjectCatalyst and TE capabilities are provided by Process Weaver.

  Our approach to instantiating this interface required:

- The identification of a small set of generic process programs that support individual or collaborative activities that can be instantiated from planned tasks. One of the clear successes from our work is the recognition that all processes have common characteristics which permits the development and instantiation of generic process programs. These generic process programs were derived from a generic architecture for process enactment.[ETT2-94]. We have effectively used this concept to dramatically reduce our need to develop customized process programs. In fact, in our use of a generic architecture to support process enactment, we have employed one of the key concepts of megaprogramming by 16specifying a product line of process programs for use by the Task Execution Support capability of the PSE layer.

- The development of systematic techniques for binding SEE tools and services to a process program at task execution time, rather than in an off-line build procedure.

- ***A Process Support Environment is a powerful SEE integration vehicle.***

  The main thesis of this paper is that a PSE is one of the most useful SEE integration vehicles, since two forms of integration are simultaneously at work: integration of the organization's process with the SEE, and integration of tools and artifacts within the context of the process.

5. CONCLUSION

We contend that it is becoming increasingly important for organizations - particularly megaprogramming organizations - to think of process and SEE as intimately tied. With the collaboration with the Air Force Demonstration Project team, we have gained valuable experience in integrating process and SEE and have identified a potentially profound SEE abstraction layer - the Process Support Environment - which has been the focus of this paper.

Section 3, Process-Based SEE Integration, presented the motivation for the PSE and provided a model for its structure and interfaces. The model identifies five cooperating "capability sets" within a PSE:

- Process Modeling;
- Project Planning, Monitoring and Control;
- Tactical Planning/Coordination;
- Task Execution; and
- Process Improvement Analysis.

Our work has provided experience in implementing and integrating each of these capability sets.

We also believe that our experience has yielded many pragmatic lessons learned that should prove useful to others pursuing process and SEE integration. Many of these lessons are cited in Section 4.3 of this paper.

Our priorities for future work are:

- Refine our model for the PSE layer (see Section 3.3, on page 14) as well as the approaches for applying SEE framework services to implement the layer.
- Develop minimum essential APIs for general-purpose artifact management and process state management repositories, and assess the viability of adapting current repository methods to serve these APIs.
- Continue to collaborate with both process and SEE researchers and practitioners, to help accelerate the convergence of the two disciplines.
- Continue to work with the Air Force to help them achieve the Demonstration Project's objectives and to capture experience. Specifically:
  - Actively participate in the "Process Engineering Support Team", designed to support the process users in following and improving the defined process - as well as to evolve the project's use of PSE automation capabilities;
  - Assist with the definition, instrumentation, and analysis of measurements to support process improvement; and
  - Make further improvements to existing PSE capabilities as funding permits.
- Continue to review process automation progress by other projects and commercial vendors.
- Participate in standardization efforts currently being pursued by industry groups, such as the Work Flow Management Coalition, to establish a reference model and API description for process work flow engines - described in this paper as the Task Execution Capability Set (see page 19).

REFERENCES

[Bristow95] Bristow D.J., Bulat R.G., Burton R. ***Product Line Process Development***, Proceedings Seventh Annual Software Technology Conference, Salt Lake City, UT, April 1995. (Published simultaneously with the present paper)

[Bulat95] Bulat R.G. ***Space and Warning Systems Center Domain Engineering Experiences***, Proceedings Seventh Annual Software Technology Conference, Salt Lake City, UT, April 1995. (Published simultaneously with the present paper)

[DemExp95] ***Air Force/STARS Demonstration Project Experience Report, Version 2.0 (Draft)***, CDRL Sequence A011-002D, Electronic Systems Center, AFMC, USAF, December 1994 (currently under review by the Government).

[Ett-92] Ett, W.H., R.H. Cobb, A. Kouchakdjian, ***Cleanroom Software Process Case Study: Lessons Learned from STARS Task IS-15***, IBM FSC Technical Report 85.0165, IBM Federal Systems Company, Gaithersburg, MD, June 26, 1992.

[Ett1-94] Ett, W.H., S. Becker, ***Evaluating the Effectiveness of Process Weaver as a Process Management Tool: A Case Study***, Proceedings of the Third Symposium of Assessment of Quality Software Development Tools, Washington, D.C, June 7-9, 1994.

[Ett2-94] Ett, W.H, R.H. Cobb, A. Kouchakdjian, ***Lowering the Entry Barrier to Process Execution Support***, Loral Federal Systems Internal Report, Gaithersburg, MD, October, 20, 1994.

[Ett3-94] Ett, W.H, Phillips, R.W. ***SCAI Process Definition Training Package***, Feb 1994, (available from the authors).

[Heimbigner95] Heimbigner, D., ***ProcessWall Process State Server, Demonstration Abstracts***, ARPA Software Environments Technology Conference, Arlington, Virginia, January 5-6, 1995.

[NGCR93] ***Reference Model for Project Support Environments, Version 2.0 (Draft)***, Next Generation Computer Resources, 2 September 1993.

[NIST93] ***NIST Special Publication 500-211, Reference Model for Frameworks of Software Engineering Environments*** (Technical Report ECMA TR/55, 3rd ed.), National Institute of Standards and Technology, August 1993.

[Randall89] Randall R.L. ***xSPER - An Approach for Generating Extensible Integrated Project Support Environments***, PhD Dissertation, University of California, San Diego, April 1989.

[Randall95] Randall R.L., Ekman R.G., Kent S.P. (Capt. USAF), Turner G.S. ***Integrating a SEE for Megaprogramming: Lessons Learned,*** Proceedings Seventh Annual Software Technology Conference, Salt Lake City, UT, April 1995. (Published simultaneously with the present paper)

[STSC94] Hanrahan R., Daud C., Meiser, K., Peterson J. ***Software Engineering Environment Technology Report***, Software Technology Support Center, OO-ALC/TISE, Hill AFB, Utah, April 1994.

[SchMel92] Shlaer S., Mellor S.J. *Object Lifecycles,* ***Modeling the World in States***, Yourdon Press, 1992.

[Sutton94] Sutton, S., personal communication, November 29, 1994.

[Trimble94] Trimble J. (ed.), ***STARS Program History 1983-1993 Version 1.1***, available from the STARS Program Office.

[Young91] Young, P.S., R.S. Taylor, ***Team-Oriented Process Programming***, UCI Technical Report 91-68, Department of Information and Computer Science, University of California, Irvine, CA, August 28, 1991.

AUTHOR BIOGRAPHIES

Dr. Richard L. Randall

Dr. Randall is currently working on the ARPA STARS program as the on-site lead for the Air Force/STARS Demonstration Project at Peterson AFB, Colorado. He provides consultation to the Air Force on megaprogramming technology transition issues and on long-range strategies for enabling a future product-line for the Space and Warning Systems Center. His area of technical focus on the Demo Project is the integration of the Demo Project Software Engineering Environment - notably the aspects dealing with the organization's process and artifact management.

Dr. Randall's research interests include software engineering methods and integrated project support environments (IPSEs). He has over 25 years of experience in all aspects of software engineering for large real-time systems on projects such as Gemini, Apollo, Safeguard, Sea Nymph, and B2. During this time, he has focused on the integration of methods and tools into the software process, and he has played lead role in various division-level software and systems engineering steering groups.

Dr. Randall received a BS in Mathematics from MIT and a PhD in Computer Science from UCSD. He is a member of the IEEE Computer Society and the ACM.

Mr. William H. Ett

Mr. Ett is currently working on the ARPA STARS Program, where he specializes in the development and transition of techniques for defining enactable processes, and the specification, design and development of automated process support applications. Mr. Ett's accomplishments include the design of the Cleanroom Engineering Process Assistant, the co-invention of the ProjectCatalyst front end and its generic process programming paradigm, the design of the initial LORAL STARS process support environment, and the co-development of the "STARS/SEI Process Definition Information Organizer Templates." Mr. Ett has over twenty years of experience in the development and delivery of government and military information processing and real-time systems.

Mr. Ett's research interests include the design and development of process support technology and tools to assist organizations in benefiting from process-driven software development, as well as the application of artificial intelligence techniques to support systems and software engineering.

Mr. Ett received a BS in Computer Science from the University of Maryland. He has also done graduate work in Computer Science and Operations Research.

APPENDIX: HISTORY OF PSE EVOLUTION

As a supplement to the main paper, this appendix provides a brief examination of the evolutionary path to the current Process Support Environment (PSE) and our conceptual model of the Process Support Environment overlay SEE layer. This will aid in understanding the description of the Demonstration Project PSE described in Section 4 of the paper.

We begin by discussing three predecessor STARS implementations, and we conclude by describing a representative contrasting approach being pursued by the Arcadia Project.

PREDECESSOR STARS IMPLEMENTATIONS

Starting in 1990, STARS evaluated tools and prototyped several environments and applications to support the process-driven development of software. This evolution was carried out in three phases:

- Development of a prototype Process Support System (PSS) to guide a project team in following a defined process. This phase led to the implementation of the Cleanroom Engineering Process Assistant (CEPA) [ETT-92]
- Development of a prototype PSS, using an interpretative process design tool to implement process support applications. This phase led to the implementation of the Process Weaver CEPA prototype [ETT1-94]
- Design and development of a pilot Process Support Environment (PSE), integrating the process definition, design, enactment and measurement capabilities developed on STARS [ETT2-94]. This pilot was actually used on the Air Force/STARS Demonstration Project. The resulting experience identified several required improvements, leading to the current Demonstration Project PSE.

The first two phases provided the user organizations with Process Support Systems - i.e., end-user systems that guided and supported practitioners in executing the organization's process. Process engineering activities (such as process definition and process programming) were performed by the PSS developers.

The third phase, yielding the current Air Force/STARS Demonstration Project toolset, provided the user organization with a full Process Support Environment (PSE) - which allowed the organization both to perform its own process engineering activities and to produce and evolve its own PSS as well.

We now believe the ability for an organization to perform a full range of process engineering activities is essential to megaprogramming.

CLEANROOM ENGINEERING PROCESS ASSISTANT (90-94)

Implementing the CEPA prototype provided the Loral STARS team with its first experience in process integration. CEPA was implemented using the KI Shell[17]. KI Shell provided a method editor to support the design of a process support system, and provided extensive library services to support the implementation of the final system using the C programming language. The resulting CEPA process support system was not easily modifiable, as the work flow for the process was embedded in the application. Thus, system modification and re-compilation was required to make a change to the system. Where processes are well understood and do not have a great deal of volatility, implementing the process support system as a compiler has performance advantages.

To accommodate changes in the SEE, CEPA routines were designed to invoke UNIX shell scripts to perform designated functions. These shell scripts could be modified without affecting the system. KI Shell supported integration with presentation and data integration mechanisms. Control integration mechanisms were not available at the time of CEPA's development. Thus CEPA was not only a process support system, but contained several of the capabilities allocated to the Process Support Environment layer.

Implementation of CEPA required a complete life cycle of software development activities from specification through implementation and testing. There are no short cuts.

The initial CEPA prototype was refined and fielded for use at the US Army's Picatinny Arsenal from 1992 through 1994, where it supported the MBC upgrade project. CEPA's field test experience demonstrated that process support systems could be built to guide a team of development engineers through a defined process, while performing their work.

Although CEPA did meet user's expectations, several issues were identified from its use:

- The system as implemented had limited scope, and extending the system meant additional software design, implementation and testing.
- CEPA had poor facilities for technical task planning and dispatching. These functions were supported using a project management system, followed by the use of a spreadsheet program.
- CEPA tasks, once dispatched were impossible to kill, thus the entire work step sequence for a task had to be walked through its completion, to delete it.

[17]KI Shell is a registered trademark of UES, Inc. of Dayton, Ohio

Table 1 provides a summary of the Process Support Environment Capabilities included in CEPA.

PSE Capabilities	PSE Tools	PSE Integration
• Process Modeling	• None	• None
• Project Planning, Monitoring and Control	• None	• None
• Process Performance Management	• Limited	• Yes, an editor was provided to add black, state and clear boxes as required. However, planning tasks required support external to the tool.
• Task Execution Support	• KI Shell Enactment Engine	• Yes, the enactment engine managed all work flow activities for the program.
• Process Improvement Analysis	• KI Shell Instrumentation Facilities	• Yes, task steps could be instrumented to collect and post measurement data.
• Process/Artifact State Management	• KI Shell State Object Manager	• Yes, KI Shell used a frame information management system to maintain state and attributes on all task and data objects.
• Collaborative Development Support	• KI Shell Messaging Services	• KI Shell permitted the implementation of process programs which supported collaborative development.
• PSS Development	• KI Shell Method Editor • C Language Compiler • Oracle	• None.

Table 1: CEPA Process Support Environment Dimensions

PROCESS WEAVER CEPA PILOT (92-93)

Implementing selected key functions of CEPA as Process Weaver cooperative procedures (or process programs), gave the Loral STARS team experience using an interpretive work flow management capability. The Process Weaver CEPA prototype was implemented using Process Weaver's process program development editors, namely, the cooperative procedure editor and the work context editor. The cooperative procedure editor permitted a process engineer to define work steps to perform a task as a petri net. Actions to support petri net processing were implemented using a script language named CoShell, which provided a LISP-like capability for manipulating objects in a CoShell program.

Process Weaver's process integration mechanisms permitted the use of control integration mechanisms, namely HP SoftBench's Broadcast Message Service (BMS) to support application invocation and transaction message passing, and

Weaver BMS, a Process Weaver implementation of BMS written to support the development of process programs which support collaborative development. Presentation integration mechanisms for process programs were managed through the Process Weaver facilities, while HP SoftBench's presentation integration mechanisms were provided for SEE engineers to encapsulate applications to be directly invoked from the SEE or from task work steps.

Process Weaver did enable the suspension or deletion of work tasks without requiring the work task to be completed.

Although Process Weaver solved a few of the concerns identified during CEPA use, it did present new issues:

- Process programs implemented using the Process Weaver cooperative procedure editor were easy to develop, and easy to change. However, change management of these process programs became problematic. Further, unless the process programs were properly managed, process practitioners had the ability to modify the process programs.
- The Process Weaver facilities available for supporting task planning, dispatching and monitoring were not effective and had to be managed outside of the Process Weaver environment.
- As in CEPA, process programming of customized applications was required. The impact of this is that customized software has to be specified, designed and implemented to support the development of software.
- Process State and History is only maintained for local executing task threads, as opposed to CEPA which provided access to global task state information to support inter- and intra-task process control.

Table 2 provides a summary of the Process Support Environment Capabilities included in the Process Weaver CEPA prototype.

PSE Capabilities	PSE Tools	PSE Integration
• Process Modeling	• None	• None
• Project Planning, Monitoring and Control	• Microsoft Project	• None. Microsoft Project plans could be exported as a flat file for use by Process Weaver.
• Process Performance Management	• Limited	• None. Task dispatching could be implemented to support Process Weaver's capabilities. However, planning and tracking these tasks required support external to the tool.
• Task Execution Support	• Process Weaver Enactment Engine using the CoShell Interpreter	• Yes, the enactment engine managed all work flow activities for the program.
• Process Improvement Analysis	• KI Shell Instrumentation Facilities	• Yes, services were provided to instrument task steps to collect and post measurement data, or to invoke external programs to collect data.
• Process/Artifact State Management	• Process Weaver CoShell Facilities	• Yes, Process Weaver maintains local state data for each process program (cooperative procedure) launched. Artifact state information must be programmed into the process program for it to be maintained.
• Collaborative Development Support	• Process Weaver BMS	• Process weaver permitted the implementation of process programs which supported collaborative development.
• PSS Development	• Process Weaver Cooperative Procedure and Work Context Editors (Process Design) • CoShell language and interpreter	• Yes, CoShell is the mechanism for executing Process Weaver process programs.

Table 2: Process Weaver CEPA Prototype Process Support Environment Dimensions.

PILOT LORAL STARS PROCESS SUPPORT ENVIRONMENT (93-94)

The pilot process support environment was implemented to provide an environment to support the activities of the process activity life cycle, namely 1) the specification of software processes, 2) the design and implementation of process programs, 3) the execution of process programs, 4) the collection of measurement data and 5) the analysis of that data to support process improvement. These activities are supported by the PSE capabilities introduced in section 3.3

To support Process Modeling and Project Planning, Monitoring and Control, the STARS-developed Process Engineering and Kernel System (PEAKS)[18] was selected. PEAKS was programmed to respond to control messages sent from either HP SoftBench's BMS or through PEAKS's API. Process state and history information was maintained by Oracle.

To support Process Performance Management, a new set of capabilities had to be developed. To satisfy the requirements for this capability set, ProjectCatalyst was implemented. ProjectCatalyst provided a capability to support the delegation of processes and the planning, dispatching and tracking of tasks required to support the requirements of a delegated process. ProjectCatalyst permitted technical managers and task leaders to plan and organize tasks using a spreadsheet paradigm, and to use this same spreadsheet view to assign, dispatch and track work tasks.

ProjectCatalyst was programmed to both send and receive control messages from PEAKS via HP SoftBench to enable ProjectCatalyst to query PEAKS about planned processes and tasks, and to post actual performance data on process and task status to PEAKS. ProjectCatalyst maintained process and task state and history data in Oracle. ProjectCatalyst was also programmed to launch process programs for execution by Process Weaver, where ProjectCatalyst controlled the work to be managed by Process Weaver. ProjectCatalyst made use of Process Weaver's process integration mechanisms for launching work tasks and passing control messages between Process Weaver users.

To support Task Execution Support, Process Weaver was selected. As stated above, Process Weaver was interfaced with ProjectCatalyst, where ProjectCatalyst controlled the invocation of Process Weaver process programs and Process Weaver controlled their execution. Process programs could be instrumented to invoke SEE tools and services, collect measurements or call measurement collection agents, and to manage artifacts.

No capabilities were selected to support Process Improvement Analysis for the Pilot PSE, other than measurement data collection.

The Pilot Loral STARS PSE included several unique features:

- It made use of a generic process programming paradigm for implementing process programs. This reduced the need for customized process programming and provided greater task management flexibility for Process Performance Management.
- It provided a flexible mechanism for planning, tracking, dispatching, and canceling tasks.
- It provided consistent process state and history data among applications - made possible through data integration using an Oracle database.

From this first integration attempt, and from the essential usage experience that resulted from trial usage on the Air Force/STARS Demonstration Project, several

[18] Formerly known as Software Process Management System (SPMS); PEAKS is ccPE's commercial name for the product.

issues were identified that needed to be addressed. Based on joint discussions of priorities, some of these issues (identified in the following list) were targeted for a major upgrade - leading to the current Demonstration Project PSE:

- Integration between PEAKS and ProjectCatalyst needed to be improved to reduce the amount of manual work required to keep the two environments in agreement. Although there were provisions for automatic reporting from ProjectCatalyst back to PEAKS, there were no provisions for automatically initializing ProjectCatalyst process information from the PEAKS database.

  (The integration is substantially improved in the current version. PEAKS exports plan data and ProjectCatalyst initializes its database accordingly.)

- Architecture services need to be developed to interface with heterogeneous SEEs to access their data and employ their services. Standard programs for providing wrappers around said services are required to facilitate application program integration into process program workflow activities. Basic problems also lie in data/file/program object ownership and poorly understood access protocols.

- Data predetermination for tasks is not sensible. The task planner is forced to anticipate data requirements to support the task prior to it being dispatched. Data objects must be assignable, either prior to task dispatching or during task execution. Further, the task performer must have access to all relevant data objects to support task work.

- Execution requirements for PSE components were too complex for the average user. User front-end programs are required to be developed to perform all housekeeping chores for PSE users.

  (A "front-end" script has now been provided to handle this automatically.)

- PSE administration is enormously complex. PSE administration programs are required to ease the burden on site personnel for both PSE installation and PSE administration.

  (A set of administrative scripts is now available that greatly simplifies administration.)

Table 3 provides a summary of the Pilot Loral STARS Process Environment Capabilities.

PSE Capabilities	PSE Tools	PSE Integration
• Process Modeling	• PEAKS	• Partially. Manual binding of plan process ids with ProjectCatalyst process pages. Automatic data reporting once binding was established through BMS.
• Project Planning, Monitoring and Control	• PEAKS • CAT/Compass (Eliminated)	• Partially. CAT/COMPASS was integrated with PEAKS to support PEAKS Plan Scheduling and status data exchange through BMS. PEAKS later subsumed required CAT/COMPASS functionality.
• Process Performance Management	• ProjectCatalyst Prototype	• Partially. Mechanism for manually binding PEAKS process ids with ProjectCatalyst process pages. Automatic data reporting once binding was established through BMS. Mechanism for invoking Process Weaver process programs.
• Task Execution Support	• Process Weaver Enactment Engine using the CoShell Interpreter	• Yes, the enactment engine managed all work flow activities for the program.
• Process Improvement Analysis	• Process Weaver CoShell Facilities • Amadeus Measurement System (AMS) • PEAKS Measurement Quality Facility (MQF)	• Yes, services were provided to instrument task steps to collect and post measurement data, or to invoke external programs to collect data, such as the Amadeus Measurement System or the PEAKS MQF.
• Process/Artifact State Management	• Oracle • Process Weaver CoShell Facilities	• Yes, Oracle was used as a persistent state object store for both PEAKS and ProjectCatalyst. • Process Weaver maintains local state data for each process program (cooperative procedure) launched. Artifact state information must be programmed into the process program for it to be maintained.
• Collaborative Development Support	• Process Weaver BMS	• Process Weaver permitted the implementation of process programs which supported collaborative development.
• PSS Development	• Process Weaver Cooperative Procedure and Work Context Editors (Process Design) • CoShell language and interpreter	• Yes. The Process Weaver editors support code and go development of process programs.

Table 3: Pilot Loral STARS Process Support Environment Dimensions.

A CONTRASTING APPROACH

ARCADIA PROCESS SUPPORT ENVIRONMENT

The Arcadia Consortium has developed a unique set of capabilities to provide process integration capabilities for their SEE development efforts, as well as provide capabilities to develop process programs. We have chosen to discuss Arcadia's Process Support Environment capabilities, because of their natural fit within the capability sets we have defined for characterizing a Process Support Environment layer.

To support Process Modeling, a tool called Teamware [Young91] was developed. Teamware is also a process specification and programming system, but could be used to support the development of process models for implementation as APPL/A programs.

To support Project Planning, Monitoring and Control, a tool called ManLobbi was developed to support project planning. ManLobbi also supports the capabilities of Process Performance Management, by supporting detailed task planning and task dispatching.[Sutton94].

To support Task Execution Support, process programs must be implemented in the APPL/A, a capability for precompiling APPL/A process programs into Ada. The resulting compiled Ada programs support process program execution.

As uniform maintenance of process state and history is important to support and coordinate the execution of many different process programs, Arcadia developed a capability called the Processwall Process State Server [Heimbigner95] to provide for the storage of process states, as well as for operations for defining and manipulating the structures of those states.

This discussion does not include all of the components of the Arcadia SEE, but addresses the relevant key components. It should be noted that these components are the result of University research and as such are not commercially available.

Table 4 provides a summary of the Process Support Environment Capabilities of the Arcadia SEE.

PSE Capabilities	PSE Tools	PSE Integration
• Process Modeling	• Teamware	• None with other Arcadia process tools.
• Project Planning, Monitoring and Control	• ManLobbi (developed for APPL/A use) • Teamware	• ManLobbi - Yes. Integrated to capture status from executing process programs.
• Process Performance Management	• ManLobbi	• Yes. Integrated to support the dispatching of tasks and the capture of status from executing process programs.
• Task Execution Support	• APPL/A Process Programs	• Yes. APPL/A process programs employ SEE presentation, data and control integration mechanisms.
• Process Improvement Analysis	• Amadeus Measurement System	• Yes. APPL/A programs can be instrumented to invoke external programs to collect and post data.
• Process/Artifact State Management	• Managed within APPL/A applications using either TRITON or ProcessWall.	• Yes. APPL/A programs can be programmed to employ the services of an object manager (TRITON) or ProcessWall to maintain process state data.
• Collaborative Development Support	• Process Weaver BMS	• Process weaver permitted the implementation of process programs which supported collaborative development.
• PSS Development	• Teamware (process design) • APPL/A (process program development)	• Teamware - Yes. Supports code and go process programming. • APPL/A - No.

Table 4: Arcadia Process Support Environment Dimensions.

Using Process to Integrate Software Engineering Environments

7th Annual Software Technology Conference
Track 7 - Architectures
Tuesday, 11 April 1995

Dr. Richard Randall
William Ett

LORAL
Federal Systems

ARPA/STARS Program

LORAL
Federal Systems

Outline

AF/STARS SCAI

- Context
- Process-Based SEE Integration
- Air Force/STARS PSE Experience
- Conclusion

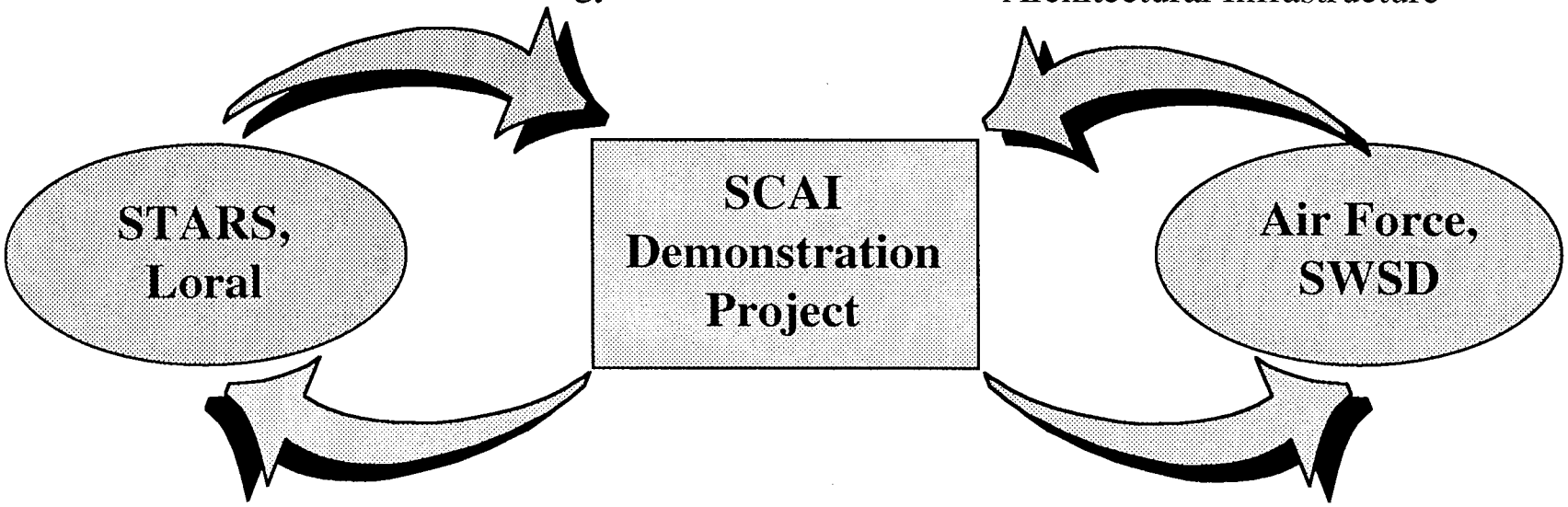
LORAL
Federal Systems

AF/STARS Demo Partnership

AF/STARS SCAI

- STARS Technology
- RISC/AIX Technology

- SWSD Technology
- Domain Expertise
- Architectural Infrastructure



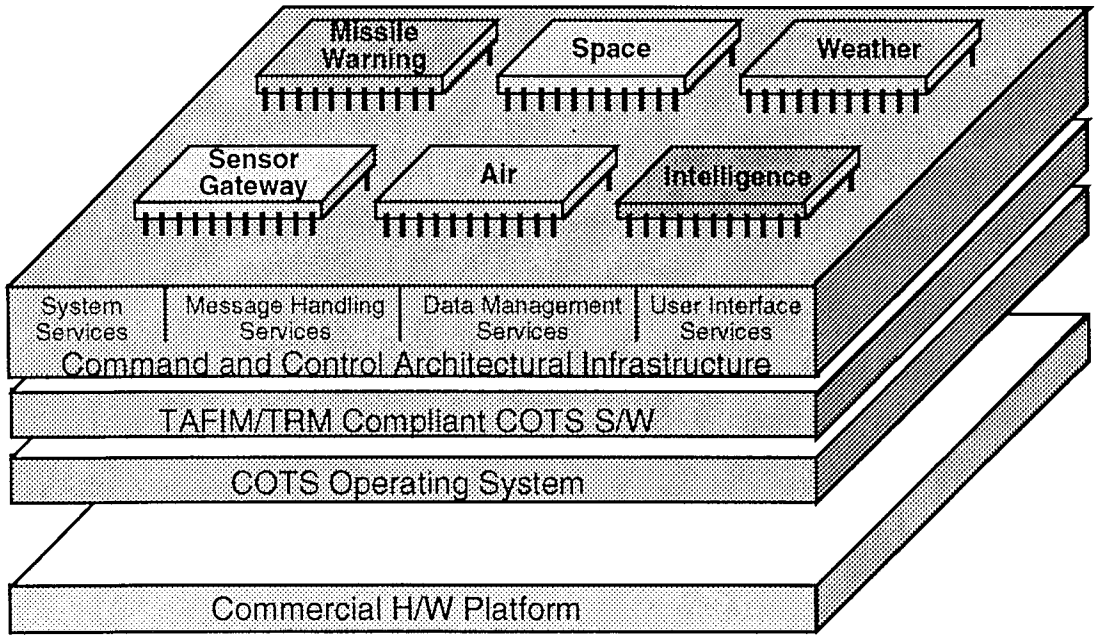
- Enhanced Technology
- Real Project Experience

- Working Application
- Product-Line Foundation

LORAL
Federal Systems

Product-Line Architecture

AF/STARS SCAI



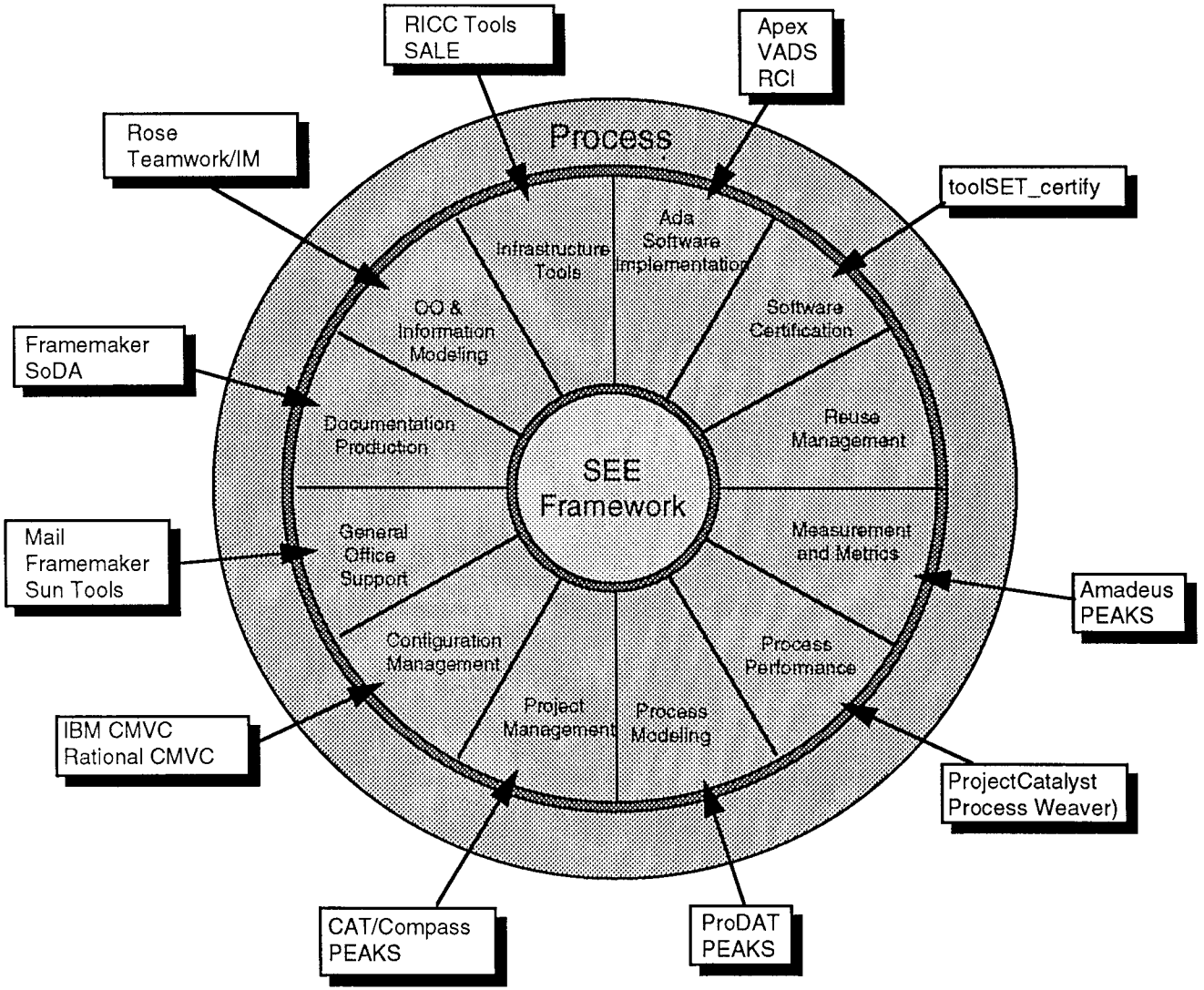
Basis for:

- Domain-Specific Reuse
- Process
- Automated Support

LORAL
Federal Systems

Demo Project SEE

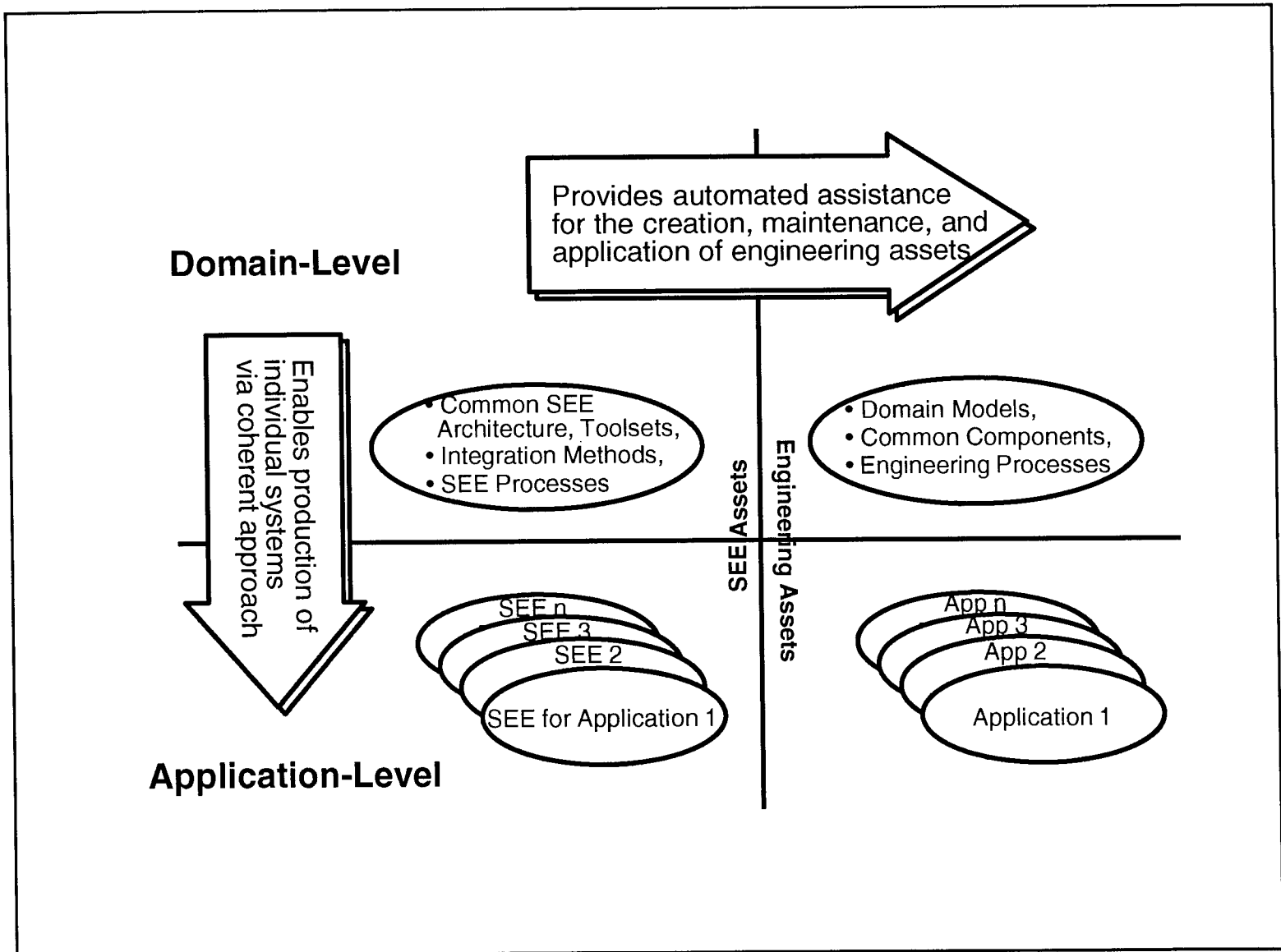
AF/STARS SCAI



LORAL Federal Systems

The SEE Product-Line

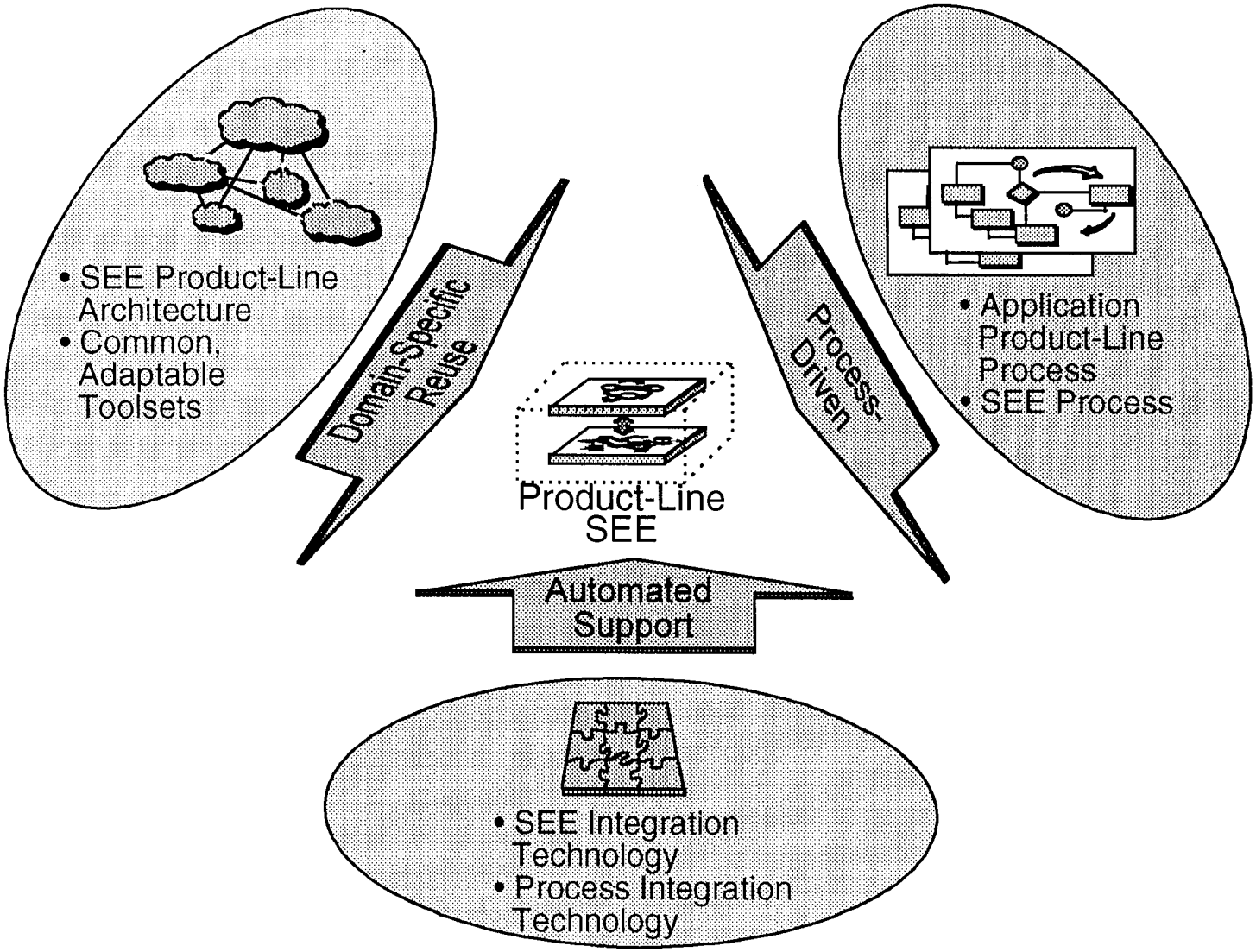
AF/STARS SCAI



LORAL
Federal Systems

Megaprogramming the SEE

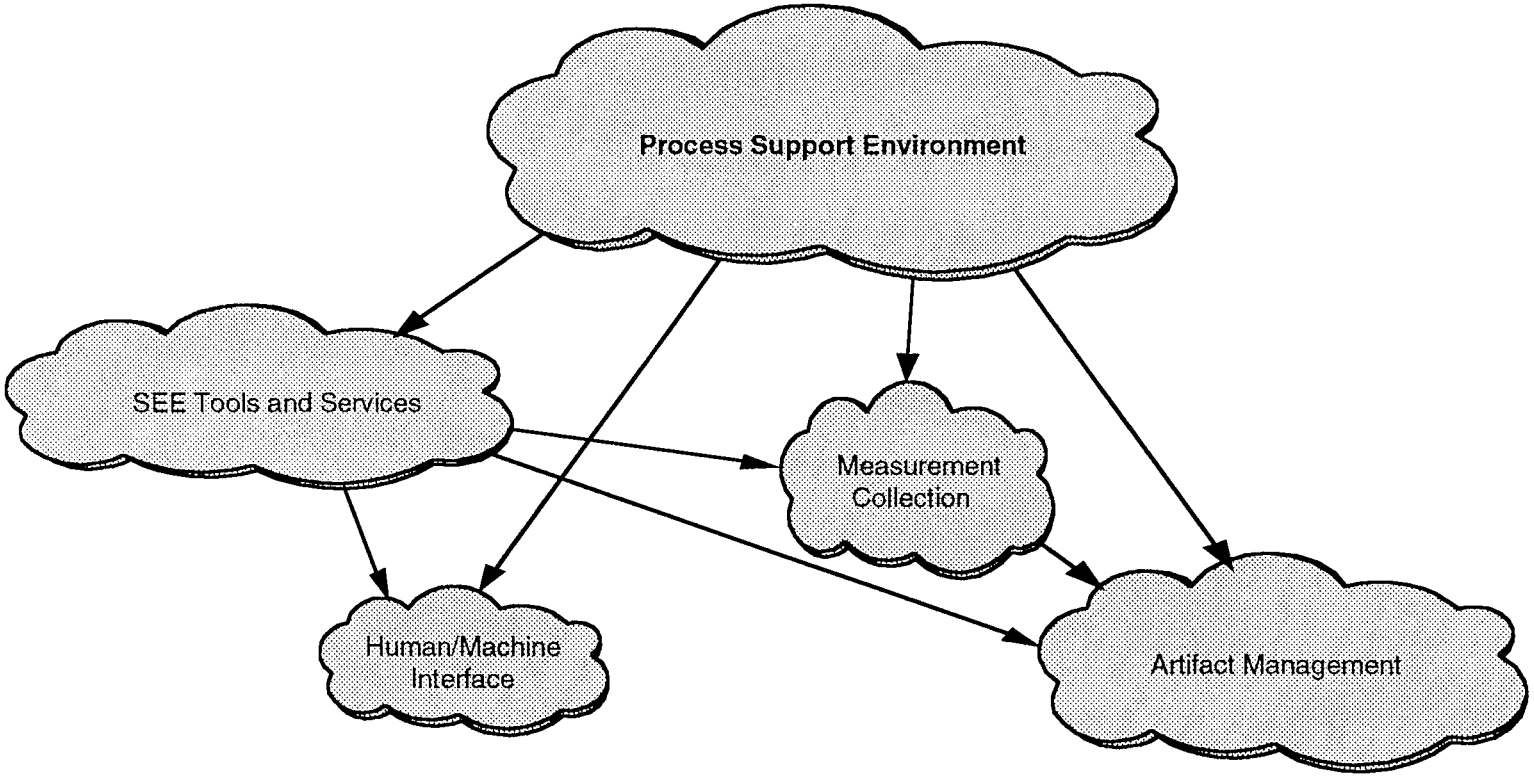
AF/STARS SCAI



LORAL
Federal Systems

SEE Architectural Layers

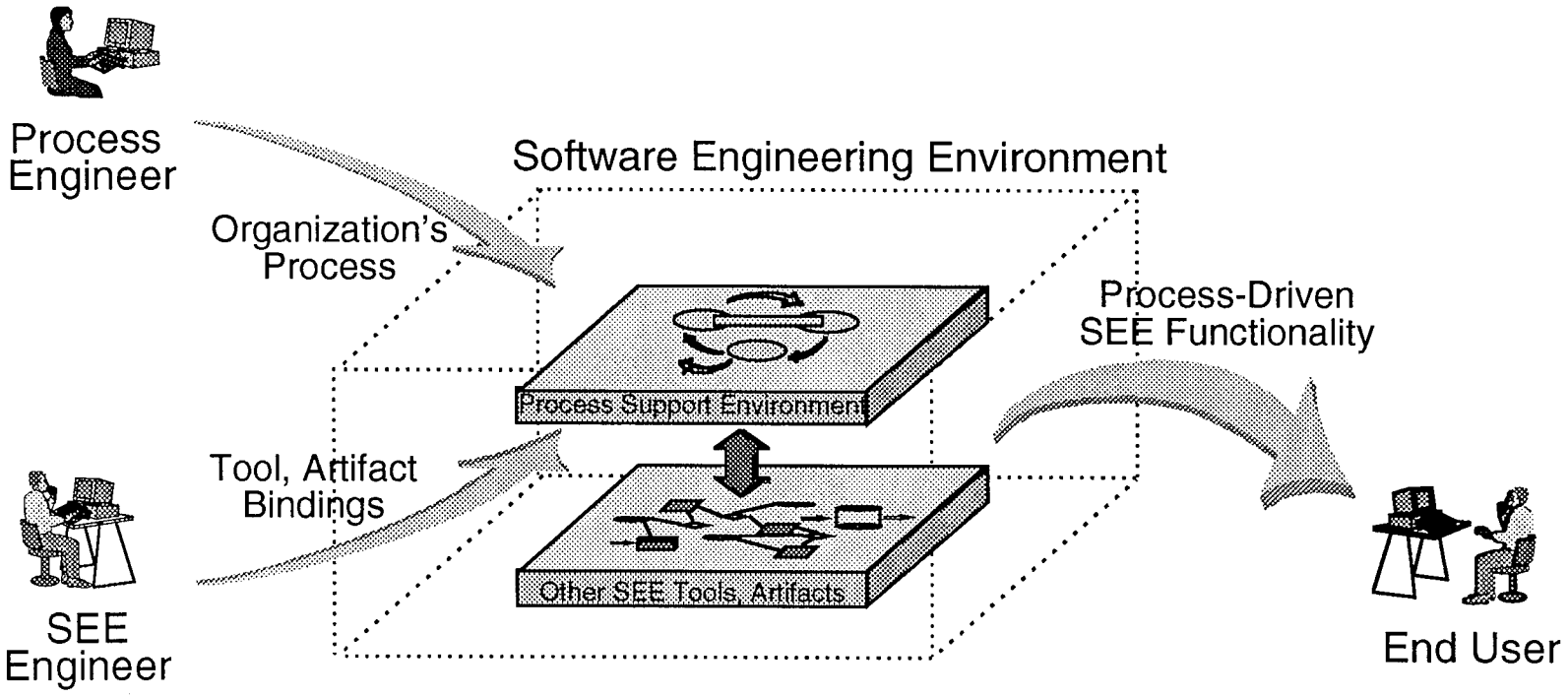
AF/STARS SCAI



LORAL
Federal Systems

Process/SEE Integration

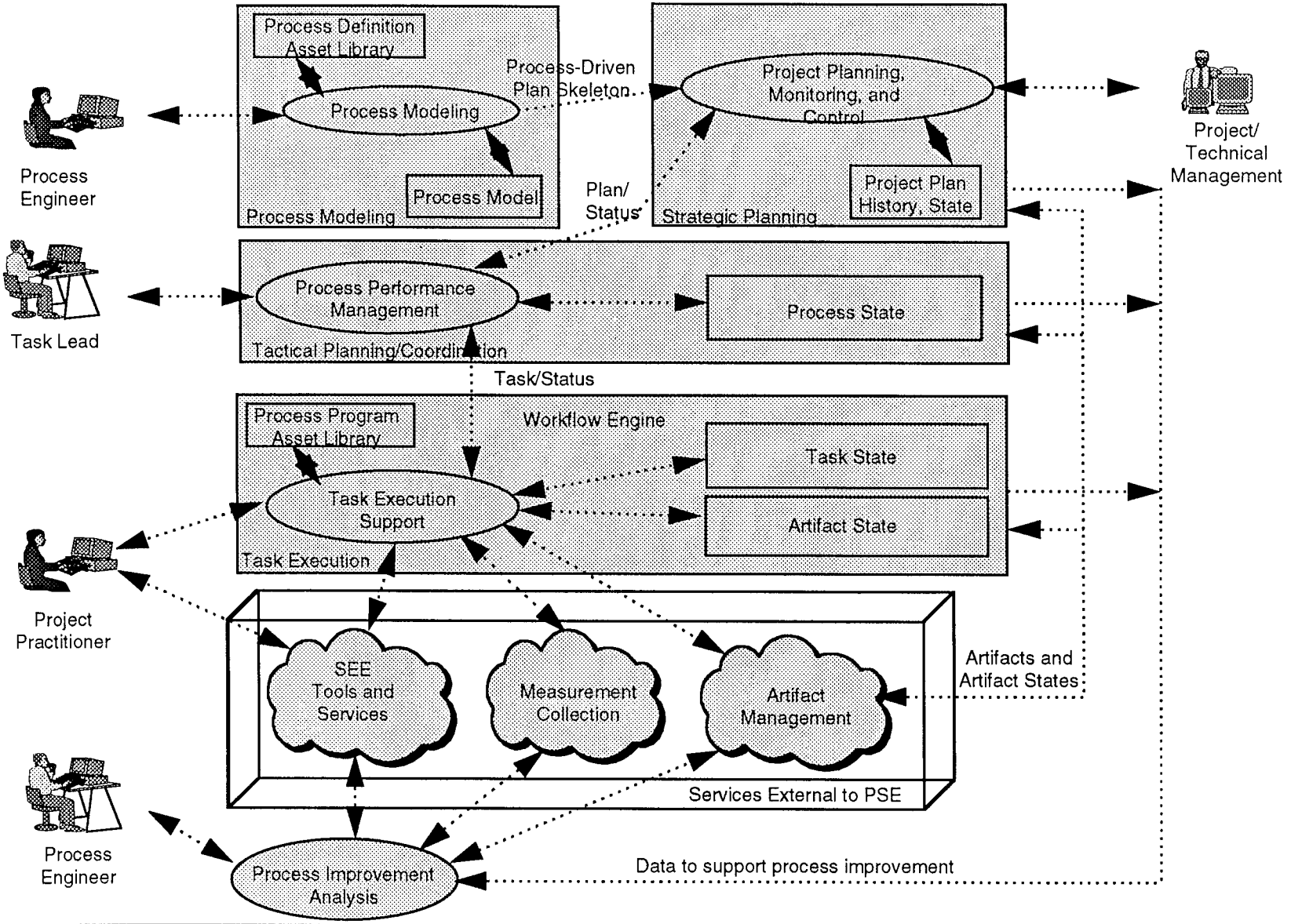
AF/STARS SCAI



LORAL Federal Systems

PSE: Structure & Interfaces

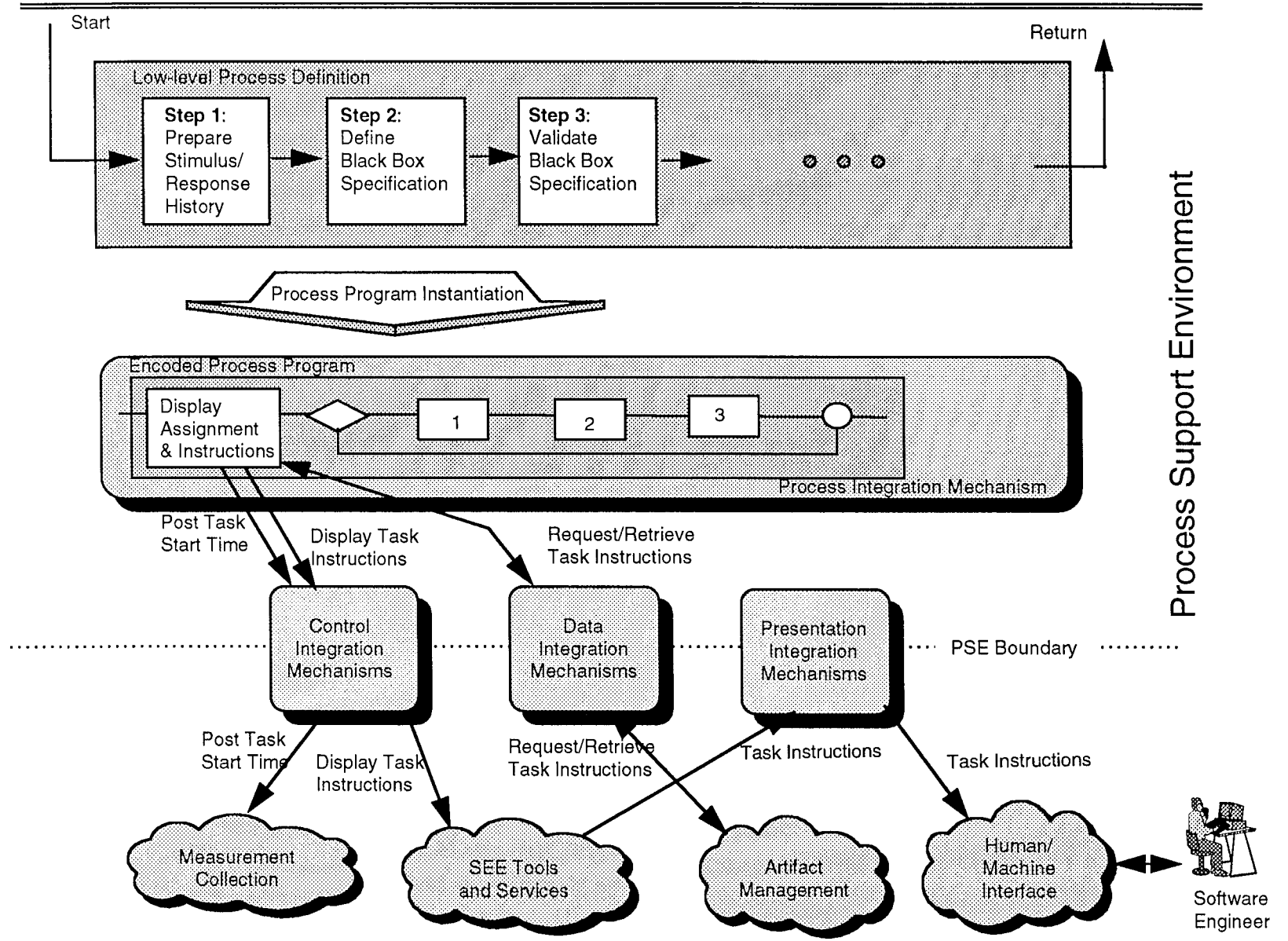
AF/STARS SCAI



LORAL
Federal Systems

Integrating the PSE with the Rest of the SEE

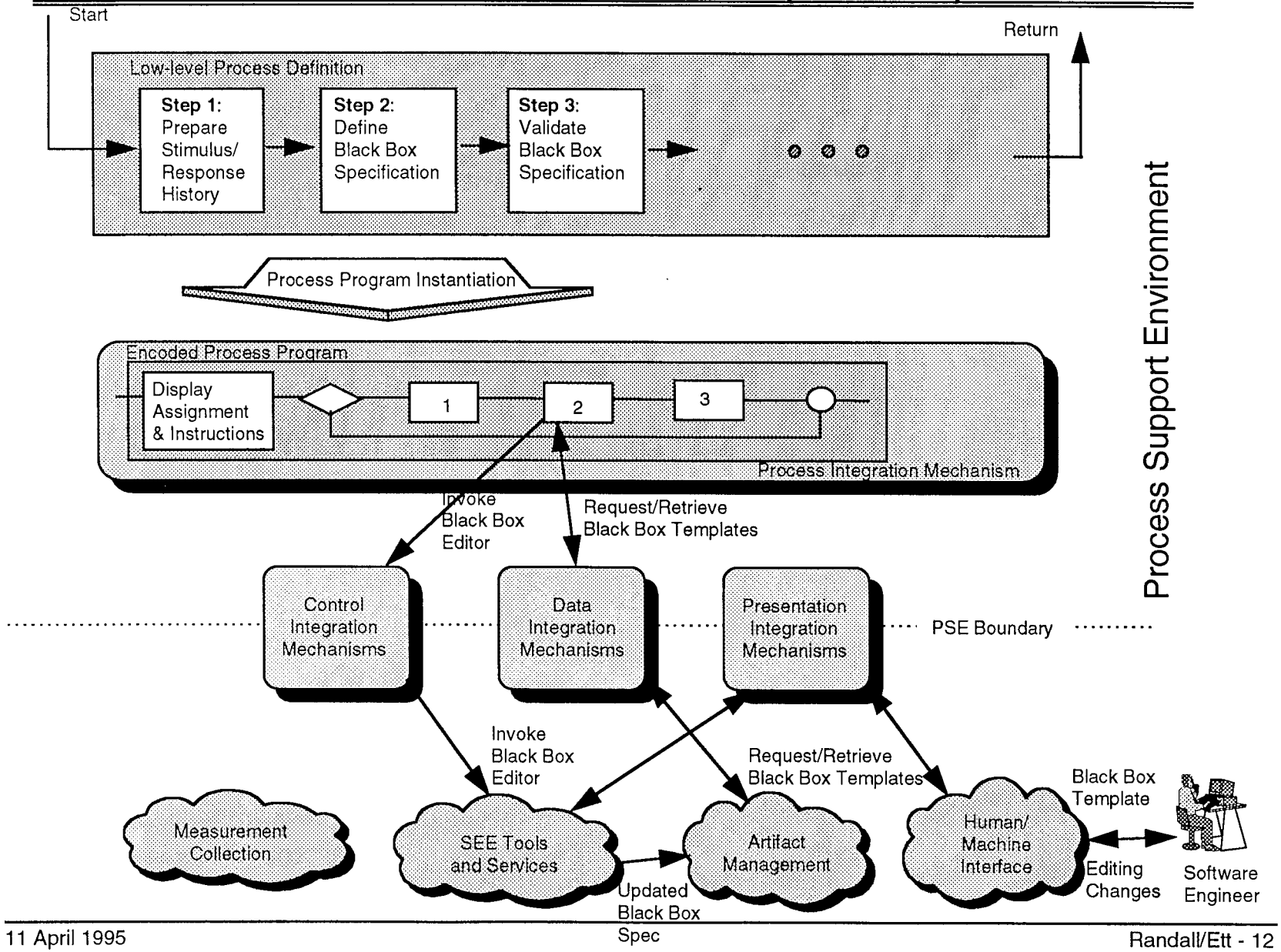
AF/STARS SCAI



LORAL
Federal Systems

Integrating the PSE with the Rest of the SEE (cont'd)

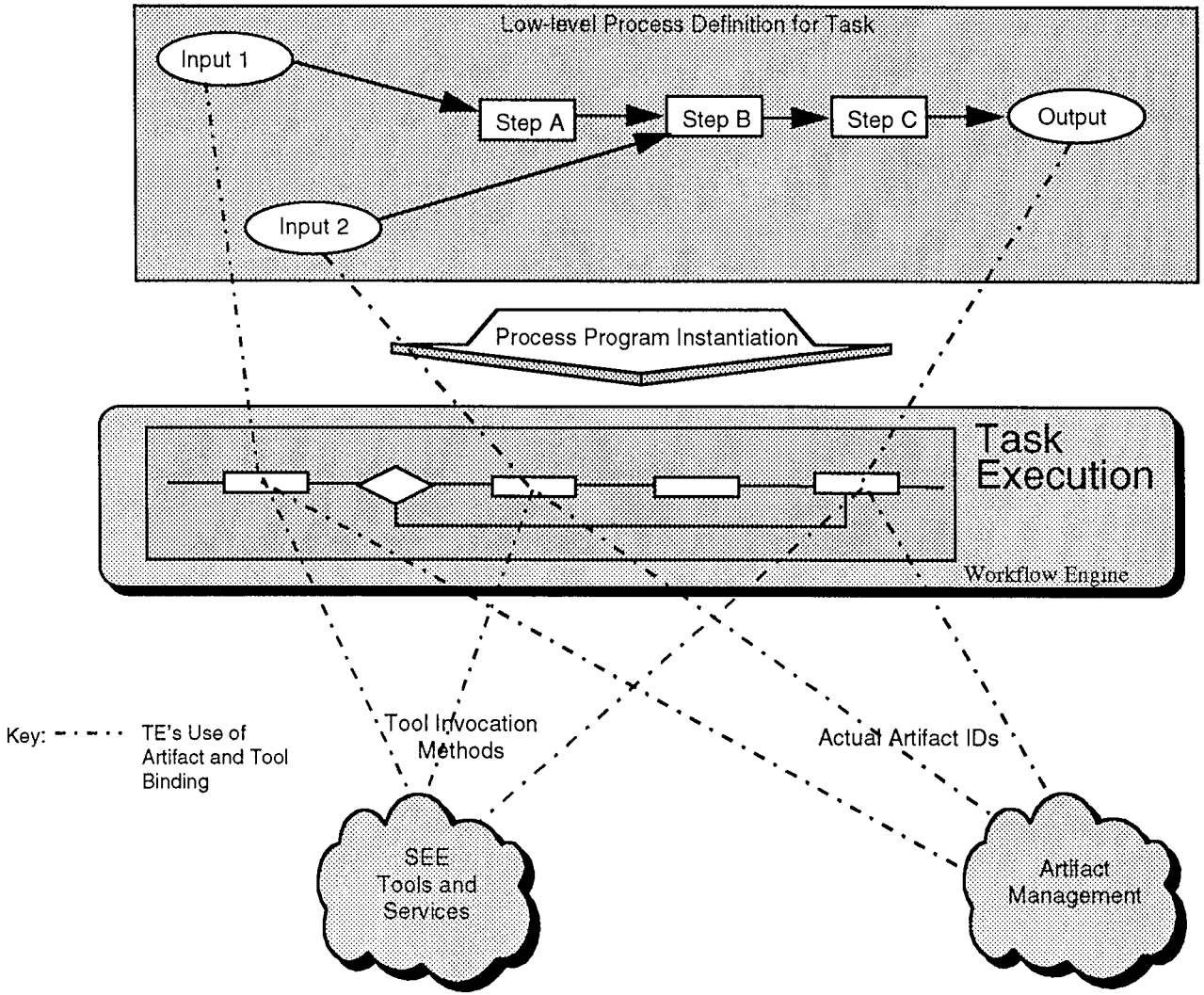
AF/STARS SCAI



LORAL Federal Systems

Binding to Tools and Artifacts

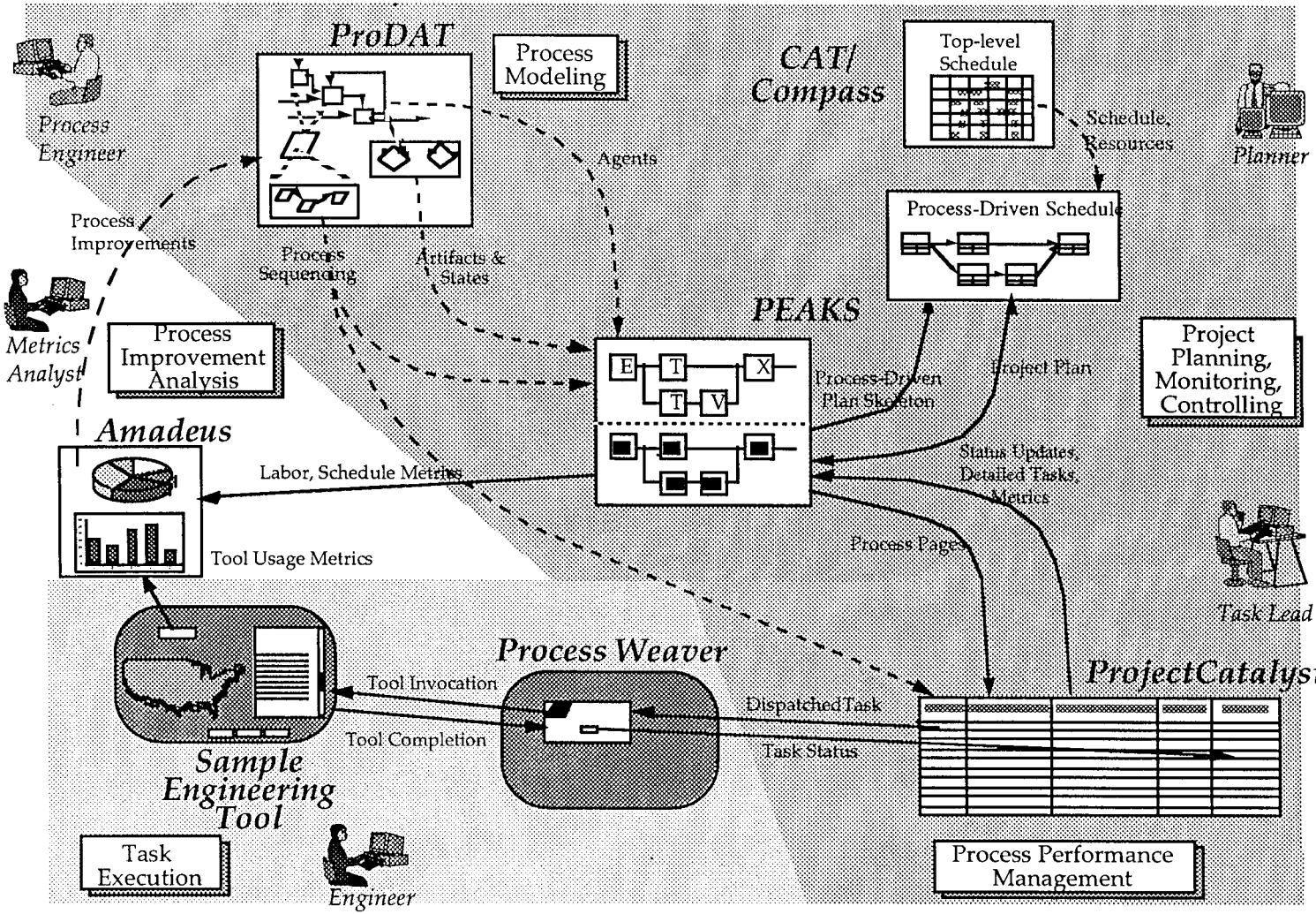
AF/STARS SCAI



LORAL
Federal Systems

The SCAI PSE

AF/STARS SCAI



Lessons - Overall

- Building a PSE: An ambitious undertaking
- The pervasive impact of fundamental change
  - New approaches affect all project areas at once
  - General rule: Incremental build-up strategy

LORAL
Federal Systems

Lessons - Integration

AF/STARS SCAI

- Paradigms require integration, too
- Subtle platform differences can confound
- Issues with broadcast messaging technology
- Pros and cons of using "off-the-shelf" functionality
- Multiple databases: a pervasive integration issue
- Monolithic artifact management by individual toolsets
- Wanted: A central artifact repository
- Wanted: A central process state repository

LORAL
Federal Systems

Lessons - PSE Innovations

AF/STARS
SCAI

- Process-driven planning
- Task management by spreadsheet
- Process programming with generics
- Tying “groupware” to process

Conclusion

- Results: Maturing understanding of process automation
  - Practical development and usage experience
  - Recognized importance of PSE encapsulation layer
  - Developed working models and taxonomy
- Current Status
- Plans
  - Refine PSE models
  - Develop APIs for Artifact and Process State repositories
  - Continue to support mutual Demo Project objectives
  - Actively participate in ongoing R&D and standardization